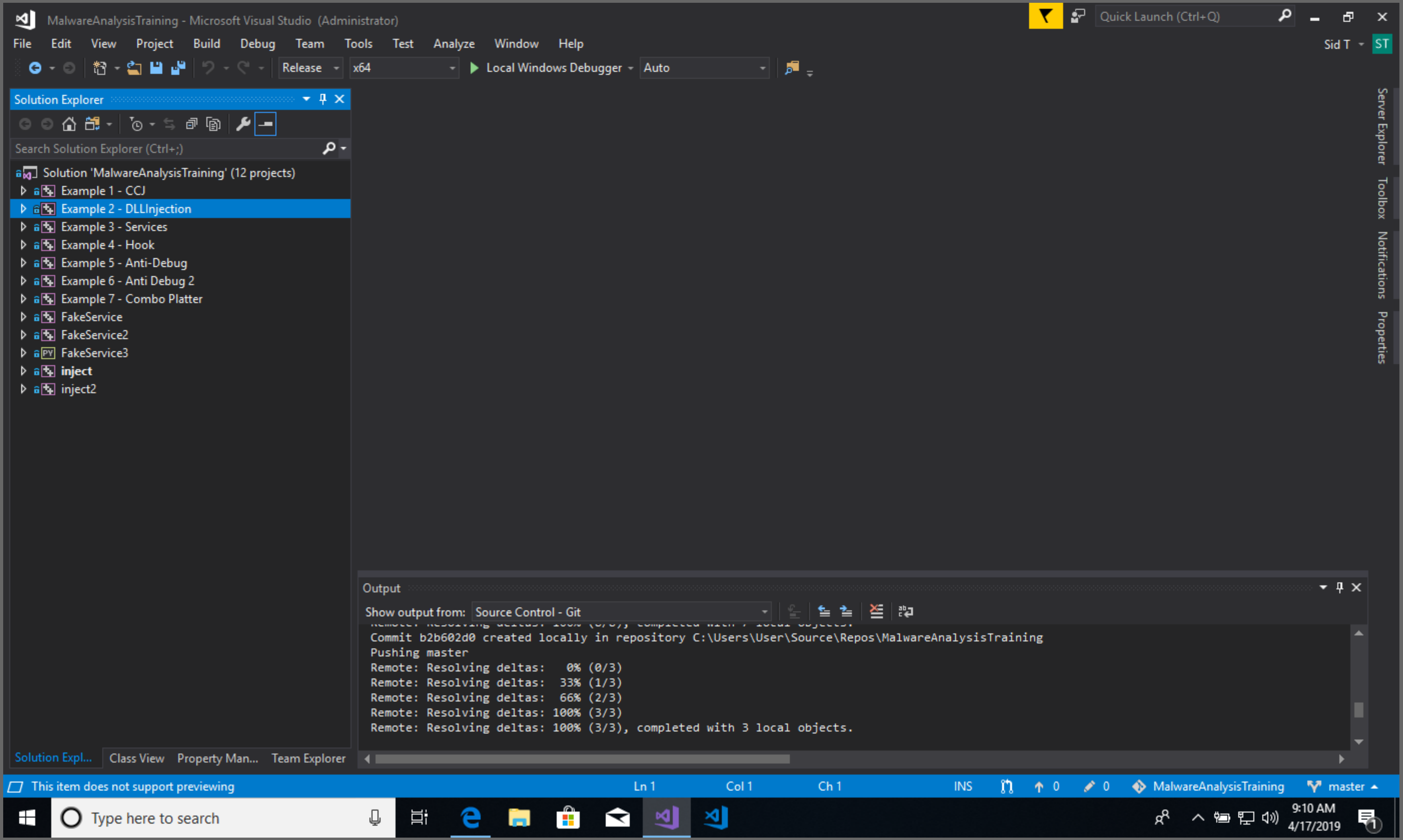
Part 1 – Example 2

Opening Comments

- This example will be more hands off and to test whether or not you remember the techniques in Example 1.
- Your goal, in this example, is to create your own simple timeline of the malware's behavior

Setting up the Example

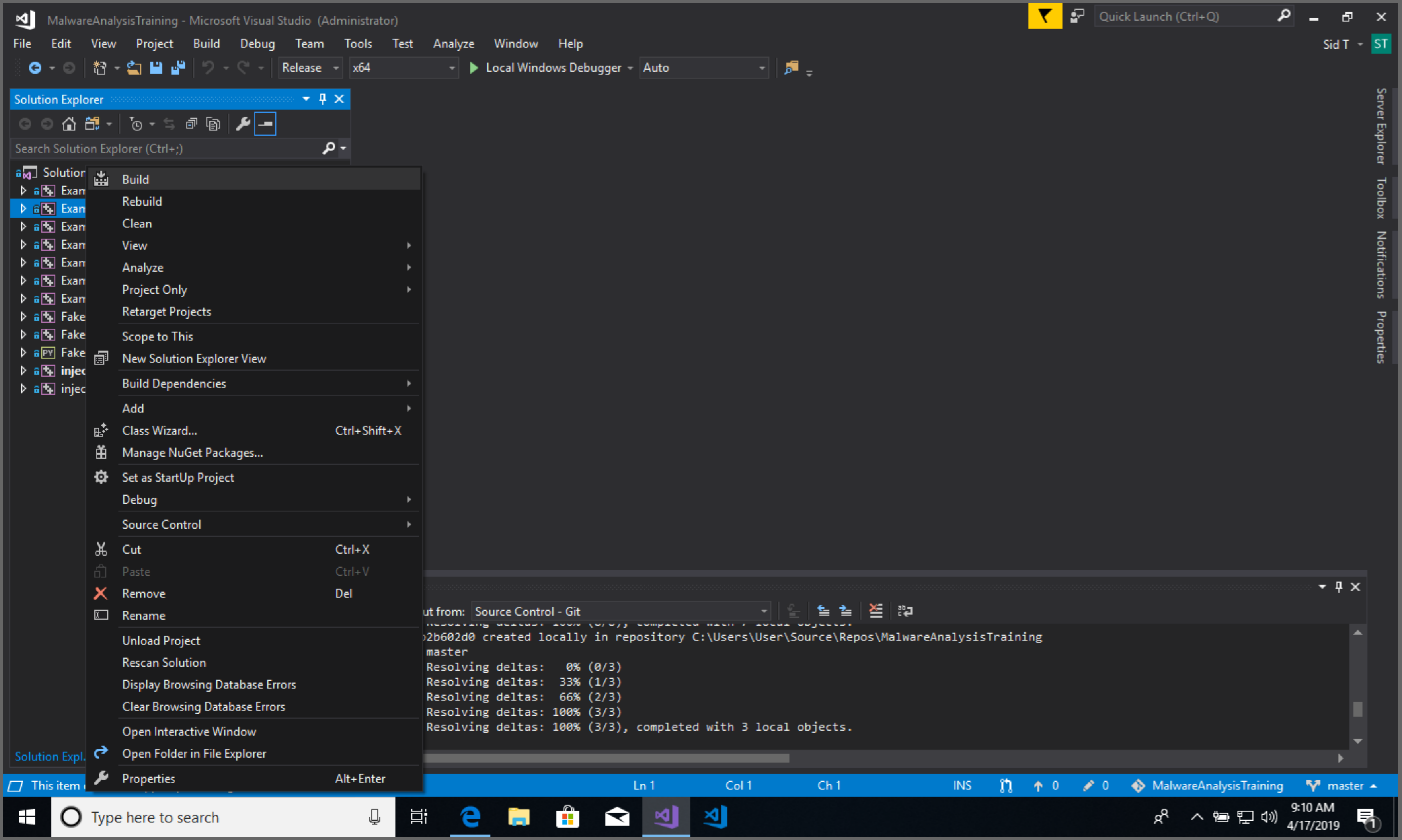


Solution 'MalwareAnalysisTraining' (12 projects)
- Example 1 - CCJ
- Example 2 - DLLInjection
- Example 3 - Services
- Example 4 - Hook
- Example 5 - Anti-Debug
- Example 6 - Anti Debug 2
- Example 7 - Combo Platter
- FakeService
- FakeService2
- FakeService3
- **inject**
- inject2

Show output from: Source Control - Git

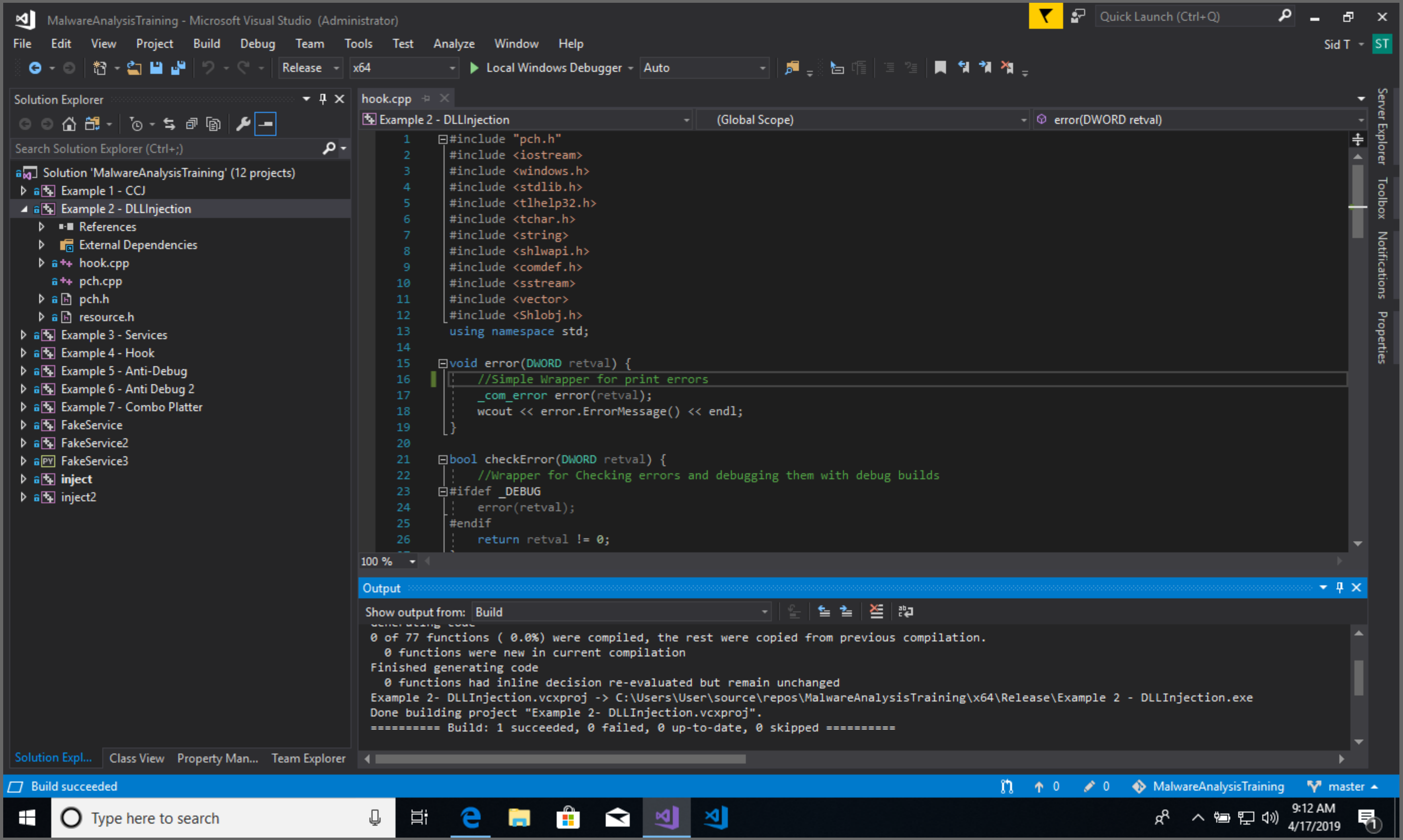
```
Commit b2b602d0 created locally in repository C:\Users\User\Source\Repos\MalwareAnalysisTraining
Pushing master
Remote: Resolving deltas:   0% (0/3)
Remote: Resolving deltas:  33% (1/3)
Remote: Resolving deltas:  66% (2/3)
Remote: Resolving deltas: 100% (3/3)
Remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
```

First things first. Build the Example 2 as a “Release” build. Don't forget to set “Any CPU” to x64/x86.



```
2b602d0 created locally in repository C:\Users\User\Source\Repos\MalwareAnalysisTraining
master
Resolving deltas:   0% (0/3)
Resolving deltas:  33% (1/3)
Resolving deltas:  66% (2/3)
Resolving deltas: 100% (3/3)
Resolving deltas: 100% (3/3), completed with 3 local objects.
```

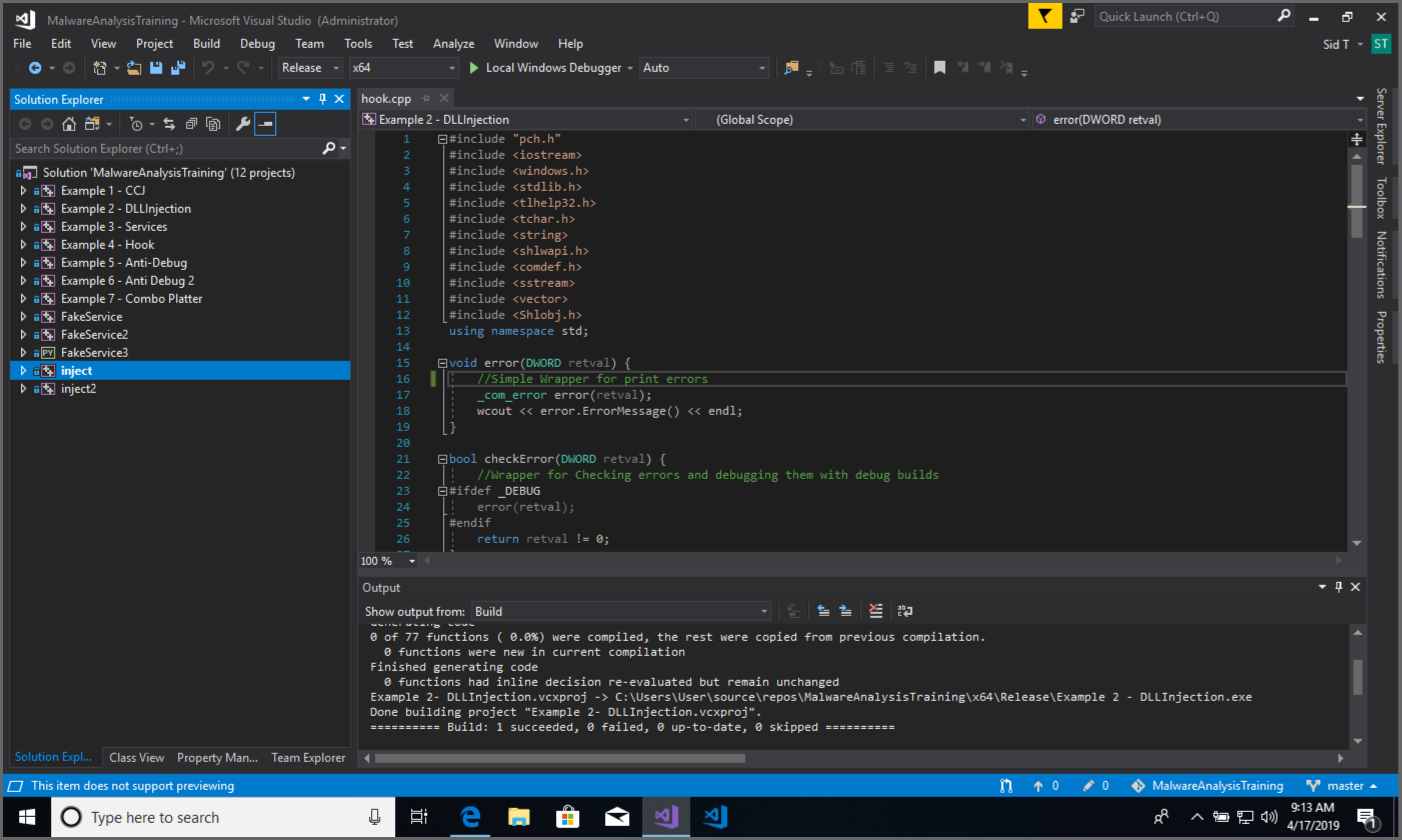
Example showing success of the build.



```
#include "pch.h"
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <tlhelp32.h>
#include <tchar.h>
#include <string>
#include <shlwapi.h>
#include <comdef.h>
#include <sstream>
#include <vector>
#include <Shlobj.h>
using namespace std;

void error(DWORD retval) {
    //Simple Wrapper for print errors
    _com_error error(retval);
    wcout << error.ErrorMessage() << endl;
}

bool checkError(DWORD retval) {
    //Wrapper for Checking errors and debugging them with debug builds
#ifdef _DEBUG
    error(retval);
#endif
    return retval != 0;
```

```
0 of 77 functions ( 0.0%) were compiled, the rest were copied from previous compilation.
  0 functions were new in current compilation
Finished generating code
  0 functions had inline decision re-evaluated but remain unchanged
Example 2- DLLInjection.vcxproj -> C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe
Done building project "Example 2- DLLInjection.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

We're also going to go ahead and build the “inject” project as well. Hover over “inject” and click the “Build” option.



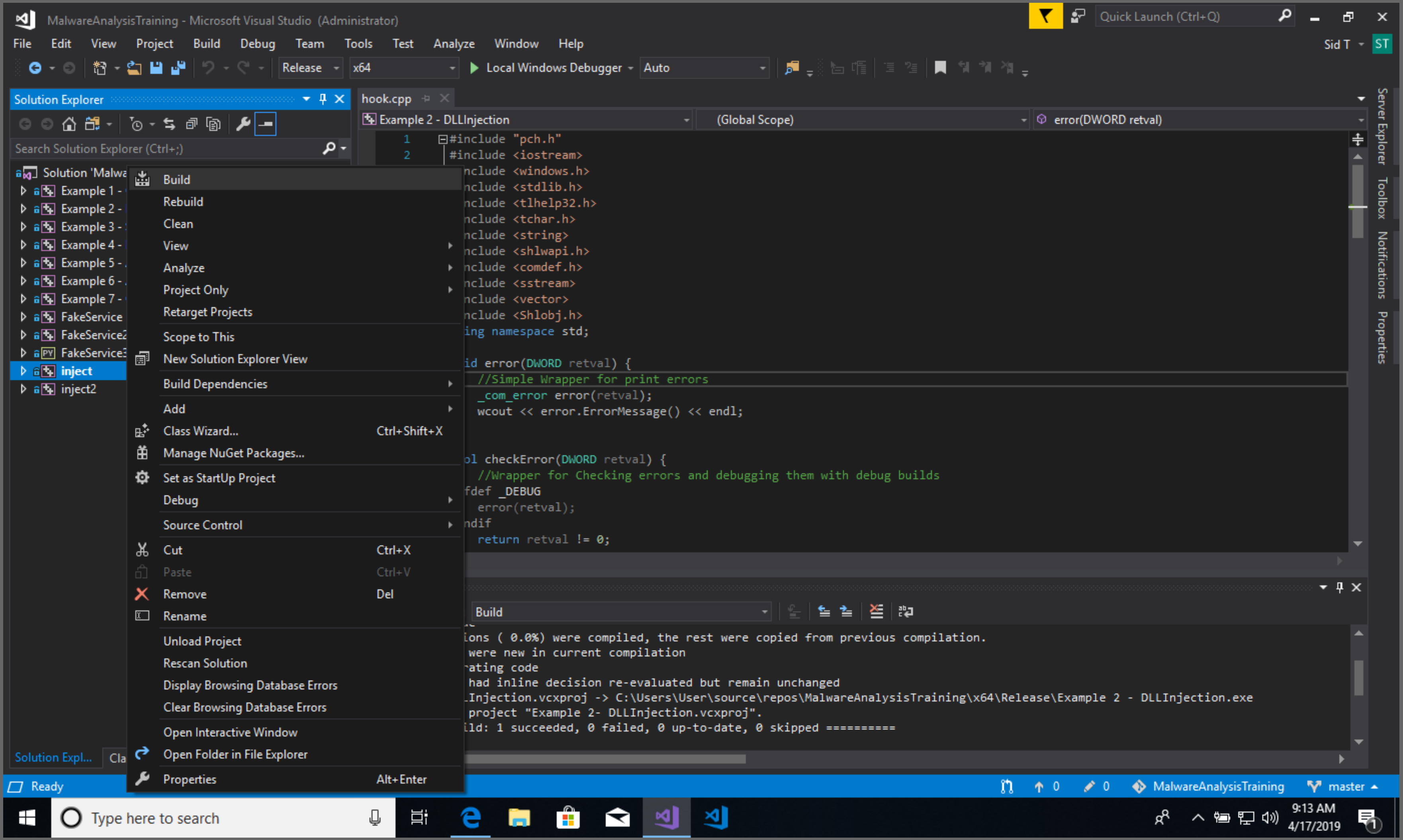
```
#include "pch.h"
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <tlhelp32.h>
#include <tchar.h>
#include <string>
#include <shlwapi.h>
#include <comdef.h>
#include <sstream>
#include <vector>
#include <Shlobj.h>
using namespace std;

void error(DWORD retval) {
    //Simple Wrapper for print errors
    _com_error error(retval);
    wcout << error.ErrorMessage() << endl;
}

bool checkError(DWORD retval) {
    //Wrapper for Checking errors and debugging them with debug builds
#ifdef _DEBUG
    error(retval);
#endif
    return retval != 0;
```

Output

Show output from: Build

```
0 of 77 functions ( 0.0%) were compiled, the rest were copied from previous compilation.
  0 functions were new in current compilation
Finished generating code
  0 functions had inline decision re-evaluated but remain unchanged
Example 2- DLLInjection.vcxproj -> C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe
Done building project "Example 2- DLLInjection.vcxproj".
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Build and success of inject.



```
#include "pch.h"
#include <iostream>
nclude <windows.h>
nclude <stdlib.h>
nclude <tlhelp32.h>
nclude <tchar.h>
nclude <string>
nclude <shlwapi.h>
nclude <comdef.h>
nclude <sstream>
nclude <vector>
nclude <Shlobj.h>
ing namespace std;

id error(DWORD retval) {
 //Simple Wrapper for print errors
 _com_error error(retval);
 wcout << error.ErrorMessage() << endl;


ol checkError(DWORD retval) {
 //Wrapper for Checking errors and debugging them with debug builds
fdef _DEBUG
 error(retval);
ndif
 return retval != 0;
```

Build

```
ions ( 0.0%) were compiled, the rest were copied from previous compilation.
 were new in current compilation
rating code
 had inline decision re-evaluated but remain unchanged
Injection.vcxproj -> C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe
 project "Example 2- DLLInjection.vcxproj".
ild: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

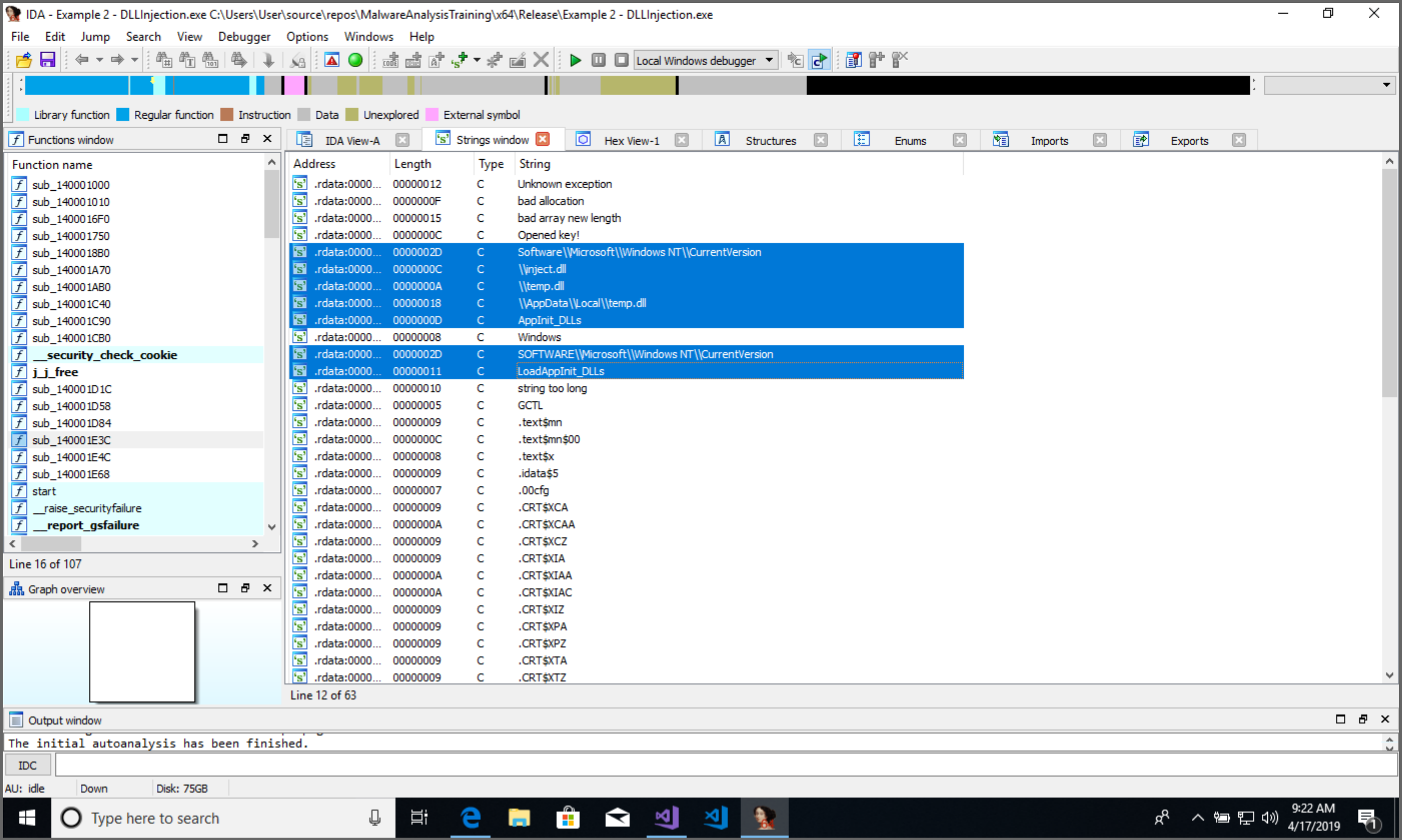
First steps

- Analyze the “Strings” first and figure out ones that are interesting.
- Come back to the slides once you've found a few and you can double check your work.
- If you get stuck look back at example 1's slides for hints.

Strings

- So your back with some strings
- In the next slide I”ll highlight from my IDA's point of view which ones I found interesting.

Strings



Function name
sub_140001000
sub_140001010
sub_1400016F0
sub_140001750
sub_1400018B0
sub_140001A70
sub_140001AB0
sub_140001C40
sub_140001C90
sub_140001CB0
__security_check_cookie
j_j_free
sub_140001D1C
sub_140001D58
sub_140001D84
sub_140001E3C
sub_140001E4C
sub_140001E68
start
__raise_securityfailure
__report_gsfailure

Line 16 of 107

Address	Length	Type	String
.rdata:0000...	00000012	C	Unknown exception
.rdata:0000...	0000000F	C	bad allocation
.rdata:0000...	00000015	C	bad array new length
.rdata:0000...	0000000C	C	Opened key!
.rdata:0000...	0000002D	C	Software\\Microsoft\\Windows NT\\CurrentVersion
.rdata:0000...	0000000C	C	\\inject.dll
.rdata:0000...	0000000A	C	\\temp.dll
.rdata:0000...	00000018	C	\\AppData\\Local\\temp.dll
.rdata:0000...	0000000D	C	AppInit_DLLs
.rdata:0000...	00000008	C	Windows
.rdata:0000...	0000002D	C	SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion
.rdata:0000...	00000011	C	LoadAppInit_DLLs
.rdata:0000...	00000010	C	string too long
.rdata:0000...	00000005	C	GCTL
.rdata:0000...	00000009	C	.text$mn
.rdata:0000...	0000000C	C	.text$mn$00
.rdata:0000...	00000008	C	.text$x
.rdata:0000...	00000009	C	.idata$5
.rdata:0000...	00000007	C	.00cfg
.rdata:0000...	00000009	C	.CRT$XCA
.rdata:0000...	0000000A	C	.CRT$XCAA
.rdata:0000...	00000009	C	.CRT$XCZ
.rdata:0000...	00000009	C	.CRT$XIA
.rdata:0000...	0000000A	C	.CRT$XIAA
.rdata:0000...	0000000A	C	.CRT$XIAC
.rdata:0000...	00000009	C	.CRT$XIZ
.rdata:0000...	00000009	C	.CRT$XPA
.rdata:0000...	00000009	C	.CRT$XPZ
.rdata:0000...	00000009	C	.CRT$XTA
.rdata:0000...	00000009	C	.CRT$XTZ

Line 12 of 63

```
The initial autoanalysis has been finished.
```

AU: idle Down Disk: 75GB

9:22 AM
4/17/2019

Strings

- Now let's discuss the ones picked out and why their interesting.
- □”Software\Microsoft\Windows NT\CurrentVersion” looks like a registry key to me and is a prime string to do analysis on.
- “temp.dll”, “inject.dll” are filenames that may be used during runtime.
- AppInit_DLLs and LoadInit_DLLs are discussed on the next slide.

AppInit_DLLs & LoadInit_DLLs

- This two are super important. Anytime you run into a string that you don't know you should use a search engine and see if a result comes up.
- If they have a Microsoft doc web page associated with them their pretty important.
- Let's see what the description is for these strings.

AppInit_DLLs & LoadInit_DLLs

- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows
- All the DLLs that are specified in this value are loaded by each Microsoft Windows-based application that is running in the current log on session.

What does it mean?

- This means that every time user32.dll is loaded into a file at runtime the file in that registry location with be loaded along with it.
- Sounds like a pretty nice spot for malware to put itself into it.

Imports

- Time to do your analysis on imports and see if there are interesting functions.
- Do some analysis and come back to the slides to double check what you've found.

Import findings.

- The ones found interesting are:
  - RegSetKeyValueA
  - RegOpenKeyExA
  - GetCurrentDirectoryA
  - MoveFileExA
  - CopyFile
- Now let's discuss why

Import Findings

- For the author of the malware to modify AppInit_DLLs they must open the registry and then edit the registry.
- RegSetKeyValueA and RegOpenKeyExA allow this to happen. You should probably put some breakpoints on these functions to test this theory.

Import Findings

  - GetCurrentDirectoryA
  - MoveFileExA
  - CopyFile
- These are indicative of file name change, file movement, and file lookup.
- The author of the malware needs to put something in place for AppInit_DLLs to work, aka a .dll file. Breakpointing here should allow us to see the file being copied and moved.

Comments on Static Analysis

- By now we can create a general hypothesis of what's happening in this file just from strings and imports.

1) The binary wishes to edit AppInit_DLLS and LoadInit_DLLs.
2) The binary wishes to move files around, possibly a .dll
3) This moved file will then be loaded later in anything that uses user32.dll

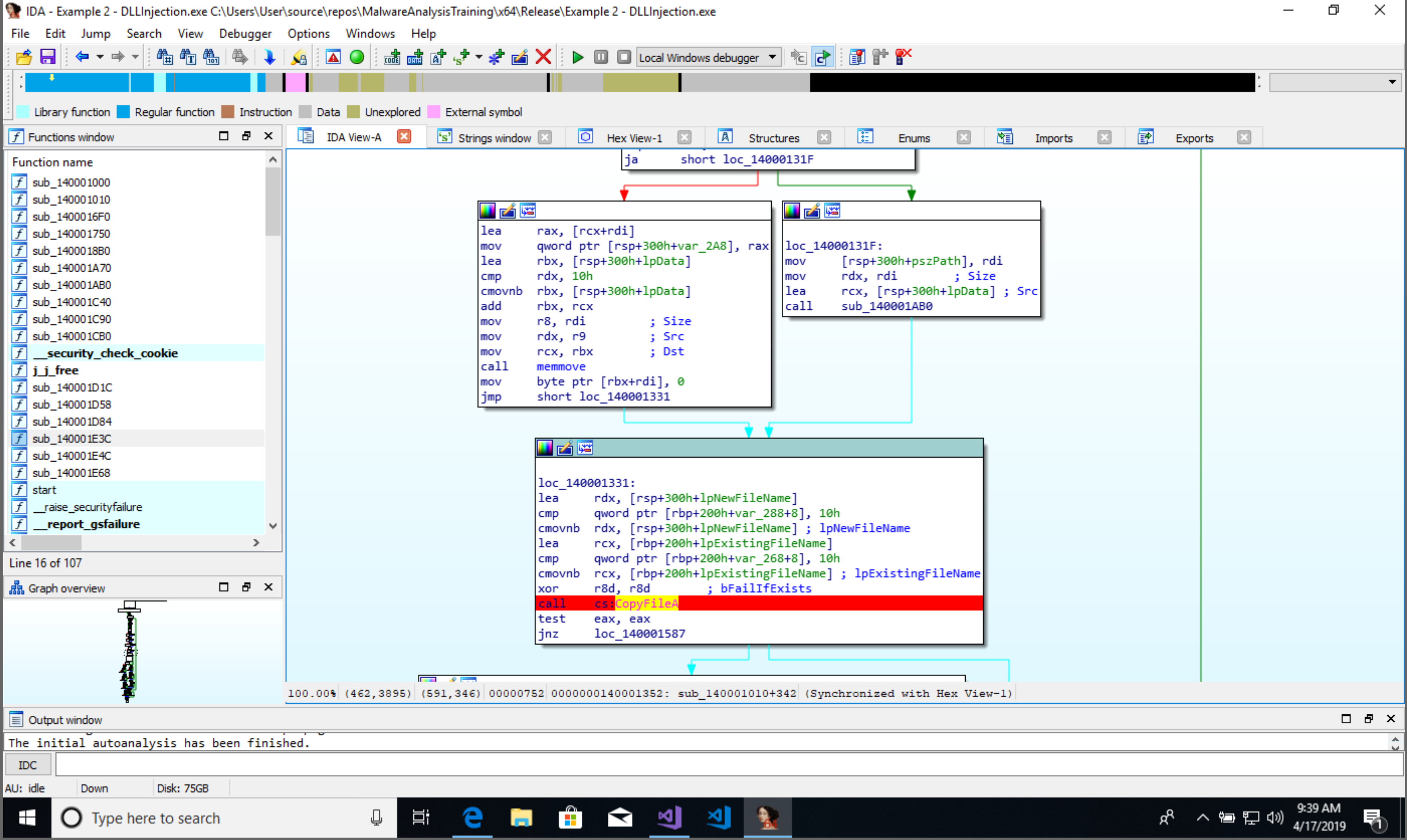
Dynamic Analysis Goal

- We have some hypothesis's to prove.
- We need to use our breakpoints and debugging skills in the dynamic analysis stage to prove these hypothesis's.
- Use the IDA debugger/Windbg to show proof that these events are happening.
- Come back to the slides once you have maybe some evidence of such things occurring

Dynamic Analysis

- Welcome back.
- What evidence did you find for our hypothesis's?
- In the next few slides I”ll show you some pictures showing some evidence of the findings and even some further things to analyze.

I stuck a breakpoint on CopyFileA to watch the input into the function.



Function name

- sub_140001000
- sub_140001010
- sub_1400016F0
- sub_140001750
- sub_1400018B0
- sub_140001A70
- sub_140001AB0
- sub_140001C40
- sub_140001C90
- sub_140001CB0
- **__security_check_cookie**
- **j_j_free**
- sub_140001D1C
- sub_140001D58
- sub_140001D84
- sub_140001E3C
- sub_140001E4C
- sub_140001E68
- start
- __raise_securityfailure
- **__report_gsfailure**

Line 16 of 107

```
ja      short loc_14000131F
```

```
lea     rax, [rcx+rdi]
mov     qword ptr [rsp+300h+var_2A8], rax
lea     rbx, [rsp+300h+lpData]
cmp     rdx, 10h
cmovnb  rbx, [rsp+300h+lpData]
add     rbx, rcx
mov     r8, rdi         ; Size
mov     rdx, r9         ; Src
mov     rcx, rbx        ; Dst
call    memmove
mov     byte ptr [rbx+rdi], 0
jmp     short loc_140001331
```

```
loc_14000131F:
mov     [rsp+300h+pszPath], rdi
mov     rdx, rdi        ; Size
lea     rcx, [rsp+300h+lpData] ; Src
call    sub_140001AB0
```

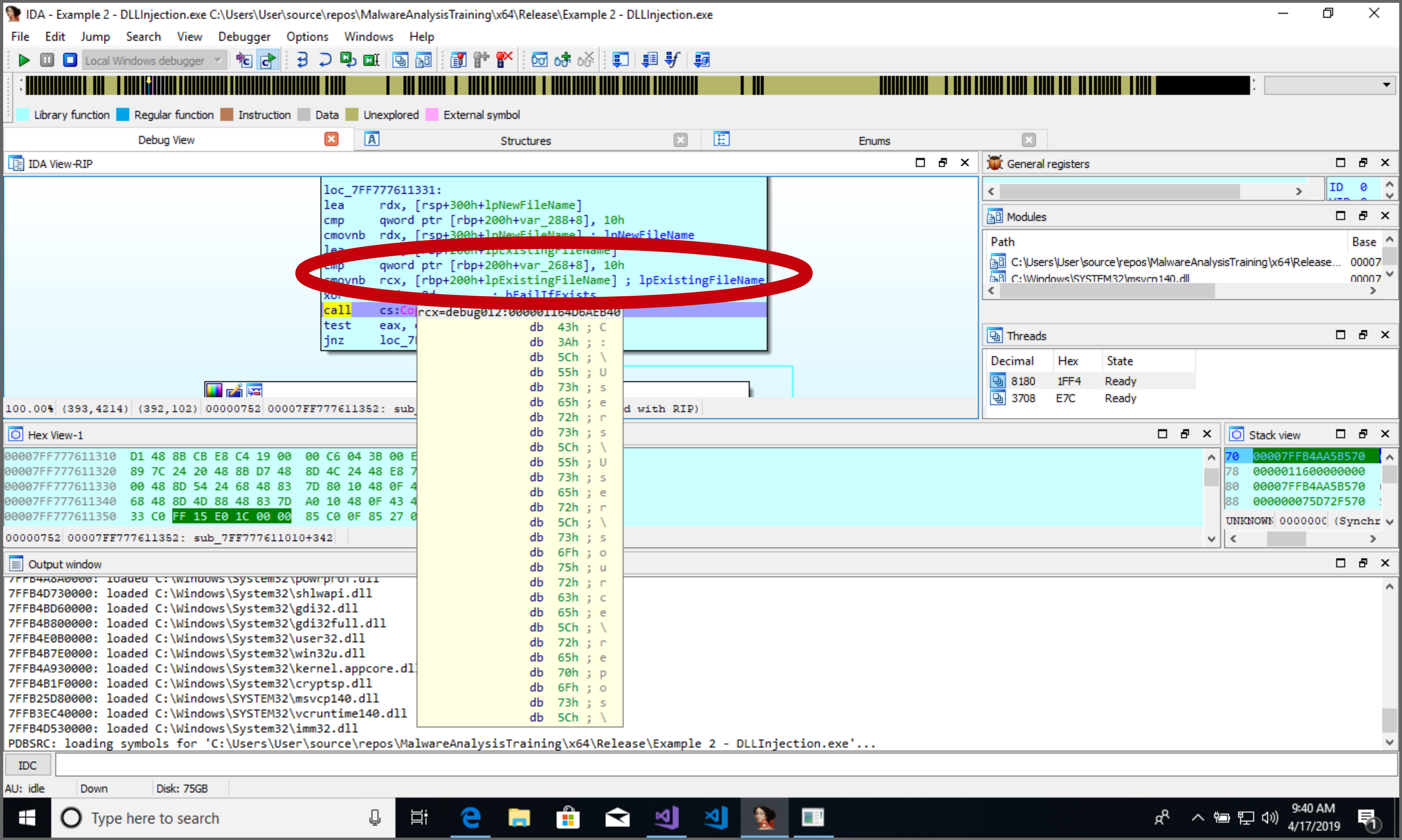
```
loc_140001331:
lea     rdx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rdx, [rsp+300h+lpNewFileName] ; lpNewFileName
lea     rcx, [rbp+200h+lpExistingFileName]
cmp     qword ptr [rbp+200h+var_268+8], 10h
cmovnb  rcx, [rbp+200h+lpExistingFileName] ; lpExistingFileName
xor     r8d, r8d        ; bFailIfExists
call    cs:CopyFileA
test    eax, eax
jnz     loc_140001587
```

```
100.00% (462,3895) (591,346) 00000752 0000000140001352: sub_140001010+342 (Synchronized with Hex View-1)
```

```
The initial autoanalysis has been finished.
```

AU: idle Down Disk: 75GB

Looks like we've got a filename going through CopyFileA. On the line circled in red there's an attribute called “lpExistingFileName” and above that “lpNewFIleName” denoted in dark blue in ida while debugging.



```
loc_7FF777611331:
lea     rdx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rdx, [rsp+300h+lpNewFileName] ; lpNewFileName
lea     rcx, [rbp+200h+lpExistingFileName]
cmp     qword ptr [rbp+200h+var_268+8], 10h
cmovnb  rcx, [rbp+200h+lpExistingFileName] ; lpExistingFileName
xor     r8d, r8d        ; bFailIfExists
call    cs:Co
test    eax, e
jnz     loc_7F
```

```
rcx=debug012:000001164D6AEB40
                    db  43h ; C
                    db  3Ah ; :
                    db  5Ch ; \
                    db  55h ; U
                    db  73h ; s
                    db  65h ; e
                    db  72h ; r
                    db  73h ; s
                    db  5Ch ; \
                    db  55h ; U
                    db  73h ; s
                    db  65h ; e
                    db  72h ; r
                    db  5Ch ; \
                    db  73h ; s
                    db  6Fh ; o
                    db  75h ; u
                    db  72h ; r
                    db  63h ; c
                    db  65h ; e
                    db  5Ch ; \
                    db  72h ; r
                    db  65h ; e
                    db  70h ; p
                    db  6Fh ; o
                    db  73h ; s
                    db  5Ch ; \
```

```
100.00% (393,4214) (392,102) 00000752 00007FF777611352: sub_     d with RIP)
```

```
00007FF777611310  D1 48 8B CB E8 C4 19 00  00 C6 04 3B 00 E
00007FF777611320  89 7C 24 20 48 8B D7 48  8D 4C 24 48 E8 7
00007FF777611330  00 48 8D 54 24 68 48 83  7D 80 10 48 0F 4
00007FF777611340  68 48 8D 4D 88 48 83 7D  A0 10 48 0F 43 4
00007FF777611350  33 C0 FF 15 E0 1C 00 00  85 C0 0F 85 27 0
00000752 00007FF777611352: sub_7FF777611010+342
```

```
ID  0
```

Path	Base
C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release...	00007
C:\Windows\SYSTEM32\msvcp140.dll	00007

Decimal	Hex	State
8180	1FF4	Ready
3708	E7C	Ready

```
70  00007FFB4AA5B570
78  0000011600000000
80  00007FFB4AA5B570
88  000000075D72F570
UNKNOWN 0000000C (Synchr
```

```
7FFB4A8A0000: loaded C:\Windows\System32\powrprof.dll
7FFB4D730000: loaded C:\Windows\System32\shlwapi.dll
7FFB4BD60000: loaded C:\Windows\System32\gdi32.dll
7FFB4B800000: loaded C:\Windows\System32\gdi32full.dll
7FFB4E0B0000: loaded C:\Windows\System32\user32.dll
7FFB4B7E0000: loaded C:\Windows\System32\win32u.dll
7FFB4A930000: loaded C:\Windows\System32\kernel.appcore.dll
7FFB4B1F0000: loaded C:\Windows\System32\cryptsp.dll
7FFB25D80000: loaded C:\Windows\SYSTEM32\msvcp140.dll
7FFB3EC40000: loaded C:\Windows\SYSTEM32\vcruntime140.dll
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
```

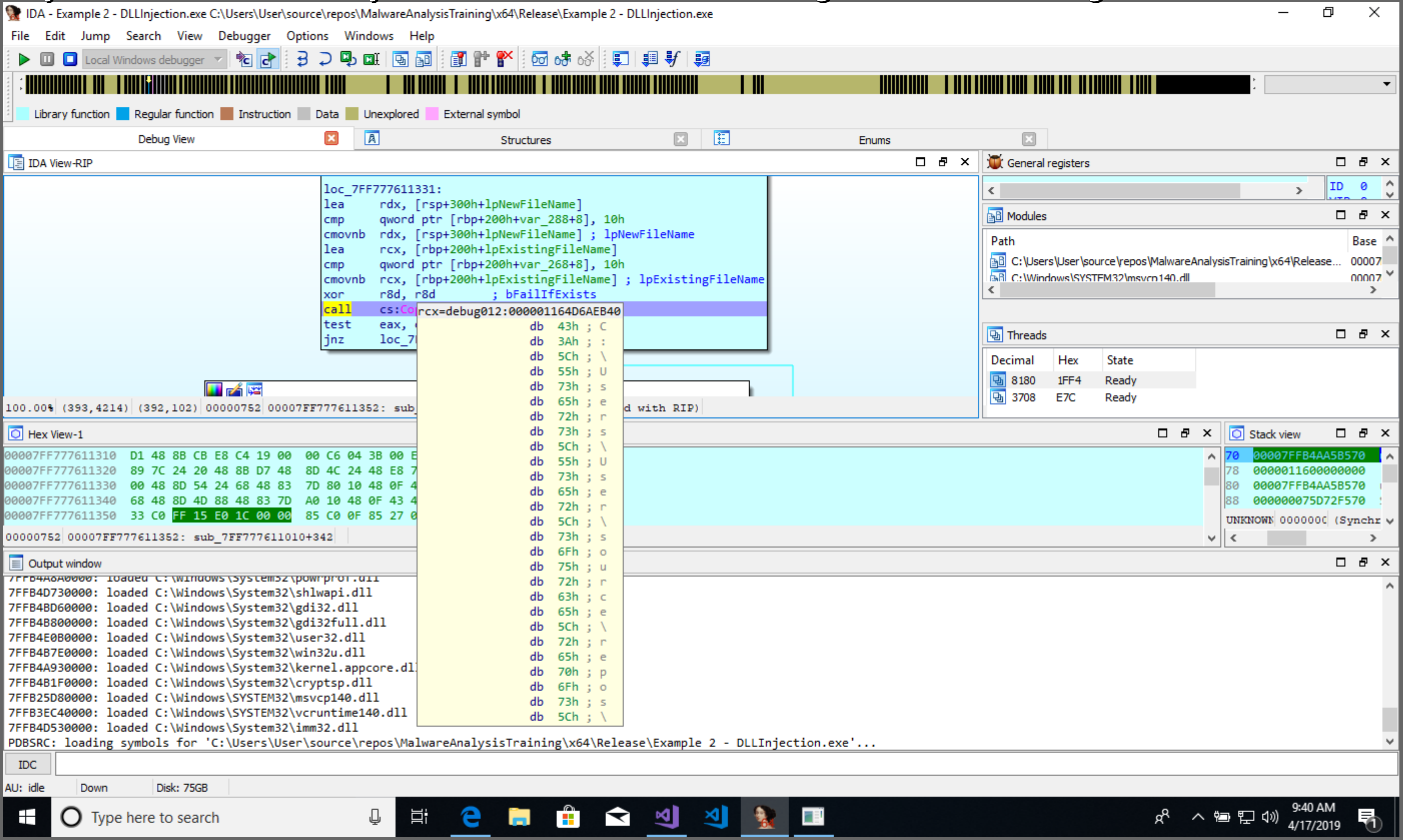
AU: idle   Down   Disk: 75GB

9:40 AM
4/17/2019

You'll notice the line has the instruction “cmovnb rcx, [rbp+200h+lpExistingFilename]”. It's a long way of saying

rcx = lpExistingFilename

If you hover over rcx you can see what the string is. It's a bit long.



```
loc_7FF777611331:
lea     rdx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rdx, [rsp+300h+lpNewFileName] ; lpNewFileName
lea     rcx, [rbp+200h+lpExistingFileName]
cmp     qword ptr [rbp+200h+var_268+8], 10h
cmovnb  rcx, [rbp+200h+lpExistingFileName] ; lpExistingFileName
xor     r8d, r8d        ; bFailIfExists
call    cs:Co
test    eax, 
jnz     loc_7
```

```
rcx=debug012:000001164D6AEB40
                    db  43h ; C
                    db  3Ah ; :
                    db  5Ch ; \
                    db  55h ; U
                    db  73h ; s
                    db  65h ; e
                    db  72h ; r
                    db  73h ; s
                    db  5Ch ; \
                    db  55h ; U
                    db  73h ; s
                    db  65h ; e
                    db  72h ; r
                    db  5Ch ; \
                    db  73h ; s
                    db  6Fh ; o
                    db  75h ; u
                    db  72h ; r
                    db  63h ; c
                    db  65h ; e
                    db  5Ch ; \
                    db  72h ; r
                    db  65h ; e
                    db  70h ; p
                    db  6Fh ; o
                    db  73h ; s
                    db  5Ch ; \
```

```
100.00% (393,4214) (392,102) 00000752 00007FF777611352: sub_ d with RIP)
```

```
00007FF777611310  D1 48 8B CB E8 C4 19 00  00 C6 04 3B 00 E
00007FF777611320  89 7C 24 20 48 8B D7 48  8D 4C 24 48 E8 7
00007FF777611330  00 48 8D 54 24 68 48 83  7D 80 10 48 0F 4
00007FF777611340  68 48 8D 4D 88 48 83 7D  A0 10 48 0F 43 4
00007FF777611350  33 C0 FF 15 E0 1C 00 00  85 C0 0F 85 27 0
00000752 00007FF777611352: sub_7FF777611010+342
```

ID 0

Path	Base
C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release...	00007
C:\Windows\SYSTEM32\msvcp140.dll	00007

Decimal	Hex	State
8180	1FF4	Ready
3708	E7C	Ready

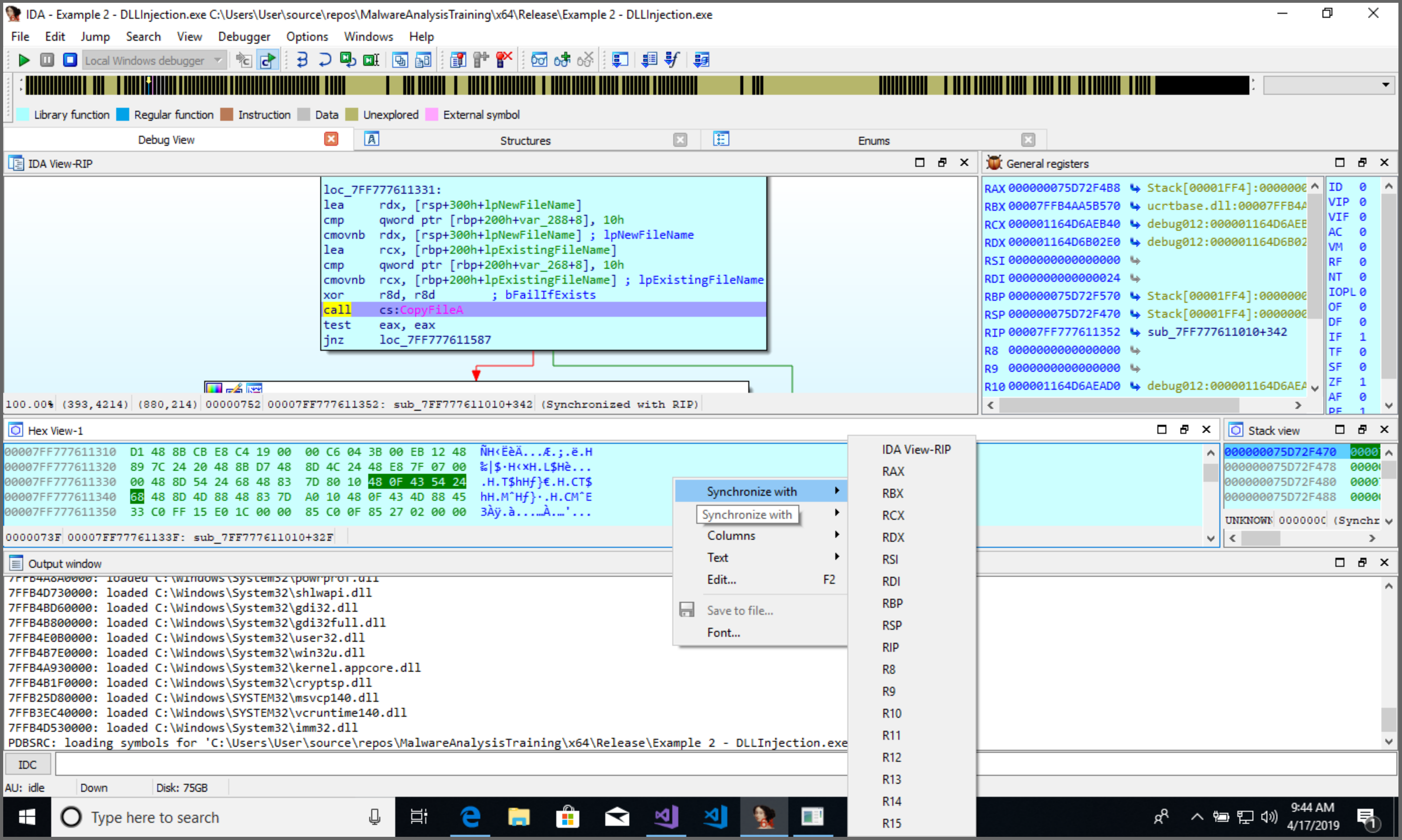
```
70  00007FFB4AA5B570
78  0000011600000000
80  00007FFB4AA5B570
88  000000075D72F570
UNKNOWN 0000000 (Synchr
```

```
7FFB4A8A0000: loaded C:\Windows\System32\powrprof.dll
7FFB4D730000: loaded C:\Windows\System32\shlwapi.dll
7FFB4BD60000: loaded C:\Windows\System32\gdi32.dll
7FFB4B800000: loaded C:\Windows\System32\gdi32full.dll
7FFB4E0B0000: loaded C:\Windows\System32\user32.dll
7FFB4B7E0000: loaded C:\Windows\System32\win32u.dll
7FFB4A930000: loaded C:\Windows\System32\kernel.appcore.dll
7FFB4B1F0000: loaded C:\Windows\System32\cryptsp.dll
7FFB25D80000: loaded C:\Windows\SYSTEM32\msvcp140.dll
7FFB3EC40000: loaded C:\Windows\SYSTEM32\vcruntime140.dll
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
```

AU: idle   Down   Disk: 75GB

9:40 AM
4/17/2019

What we do is in the “Hex view – 1” sub view, below the box graph view, is right click the hex view and you'll see a “Synchronize with” button. Hover over to RCX and click RCX to make the hex view align with RCX in memory.



IDA - Example 2 - DLLInjection.exe C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe

```
loc_7FF777611331:
lea     rdx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rdx, [rsp+300h+lpNewFileName] ; lpNewFileName
lea     rcx, [rbp+200h+lpExistingFileName]
cmp     qword ptr [rbp+200h+var_268+8], 10h
cmovnb  rcx, [rbp+200h+lpExistingFileName] ; lpExistingFileName
xor     r8d, r8d        ; bFailIfExists
call    cs:CopyFileA
test    eax, eax
jnz     loc_7FF777611587
```

```
100.00% (393,4214) (880,214) 00000752 00007FF777611352: sub_7FF777611010+342 (Synchronized with RIP)
```

```
RAX 000000075D72F4B8 ↳ Stack[00001FF4]:0000000
RBX 00007FFB4AA5B570 ↳ ucrtbase.dll:00007FFB4A
RCX 000001164D6AEB40 ↳ debug012:000001164D6AEB
RDX 000001164D6B02E0 ↳ debug012:000001164D6B02
RSI 0000000000000000 ↳
RDI 0000000000000024 ↳
RBP 000000075D72F570 ↳ Stack[00001FF4]:0000000
RSP 000000075D72F470 ↳ Stack[00001FF4]:0000000
RIP 00007FF777611352 ↳ sub_7FF777611010+342
R8  0000000000000000 ↳
R9  0000000000000000 ↳
R10 000001164D6AEAD0 ↳ debug012:000001164D6AEA
```

```
ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
IF   1
TF   0
SF   0
ZF   1
AF   0
PF   1
```

```
00007FF777611310  D1 48 8B CB E8 C4 19 00  00 C6 04 3B 00 EB 12 48  ÑH‹ËèÄ...Æ.;.ë.H
00007FF777611320  89 7C 24 20 48 8B D7 48  8D 4C 24 48 E8 7F 07 00  ‰|$·H‹×H.L$Hè...
00007FF777611330  00 48 8D 54 24 68 48 83  7D 80 10 48 0F 43 54 24  .H.T$hHƒ}€.H.CT$
00007FF777611340  68 48 8D 4D 88 48 83 7D  A0 10 48 0F 43 4D 88 45  hH.MˆHƒ}·.H.CMˆE
00007FF777611350  33 C0 FF 15 E0 1C 00 00  85 C0 0F 85 27 02 00 00  3Àÿ.à....À...'...
0000073F 00007FF77761133F: sub_7FF777611010+32F
```

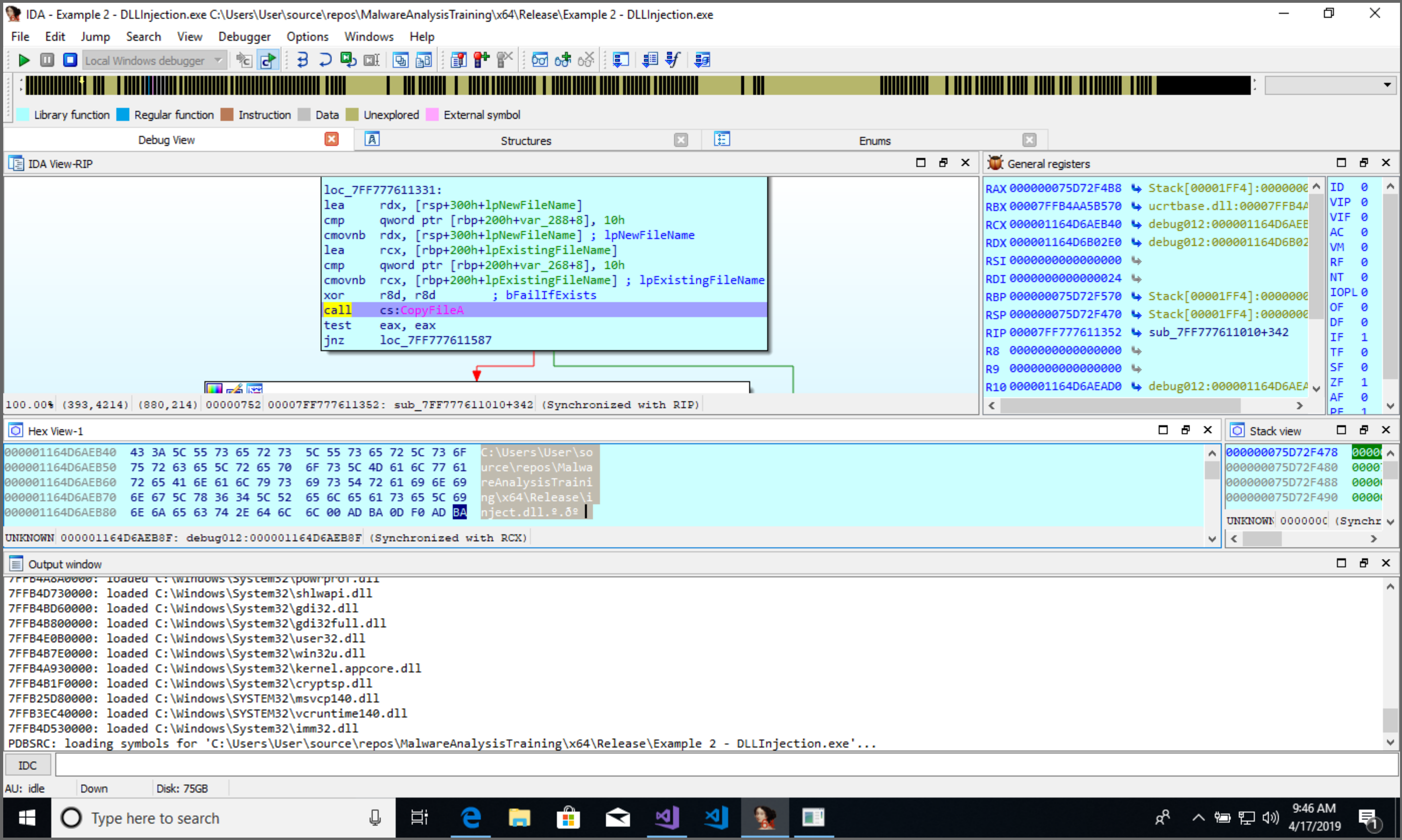
```
000000075D72F470  0000
000000075D72F478  0000
000000075D72F480  0000
000000075D72F488  0000
UNKNOWN 0000000C (Synchr
```

```
7FFB4A8A0000: loaded C:\Windows\System32\powrprof.dll
7FFB4D730000: loaded C:\Windows\System32\shlwapi.dll
7FFB4BD60000: loaded C:\Windows\System32\gdi32.dll
7FFB4B800000: loaded C:\Windows\System32\gdi32full.dll
7FFB4E0B0000: loaded C:\Windows\System32\user32.dll
7FFB4B7E0000: loaded C:\Windows\System32\win32u.dll
7FFB4A930000: loaded C:\Windows\System32\kernel.appcore.dll
7FFB4B1F0000: loaded C:\Windows\System32\cryptsp.dll
7FFB25D80000: loaded C:\Windows\SYSTEM32\msvcp140.dll
7FFB3EC40000: loaded C:\Windows\SYSTEM32\vcruntime140.dll
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe
```

AU: idle   Down   Disk: 75GB

I highlighted in gray below to show the interesting filename that pops up in memory. It looks like our inject.dll.

Now see the line with lpNewFileName? Synchronize with RDX to see what his value is.



```
loc_7FF777611331:
lea     rdx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rdx, [rsp+300h+lpNewFileName] ; lpNewFileName
lea     rcx, [rbp+200h+lpExistingFileName]
cmp     qword ptr [rbp+200h+var_268+8], 10h
cmovnb  rcx, [rbp+200h+lpExistingFileName] ; lpExistingFileName
xor     r8d, r8d        ; bFailIfExists
call    cs:CopyFileA
test    eax, eax
jnz     loc_7FF777611587
```

```
100.00% (393,4214) (880,214) 00000752 00007FF777611352: sub_7FF777611010+342 (Synchronized with RIP)
```

```
RAX 000000075D72F4B8 ↳ Stack[00001FF4]:0000000
RBX 00007FFB4AA5B570 ↳ ucrtbase.dll:00007FFB4A
RCX 000001164D6AEB40 ↳ debug012:000001164D6AEB
RDX 000001164D6B02E0 ↳ debug012:000001164D6B02
RSI 0000000000000000 ↳
RDI 0000000000000024 ↳
RBP 000000075D72F570 ↳ Stack[00001FF4]:0000000
RSP 000000075D72F470 ↳ Stack[00001FF4]:0000000
RIP 00007FF777611352 ↳ sub_7FF777611010+342
R8  0000000000000000 ↳
R9  0000000000000000 ↳
R10 000001164D6AEAD0 ↳ debug012:000001164D6AEA
```

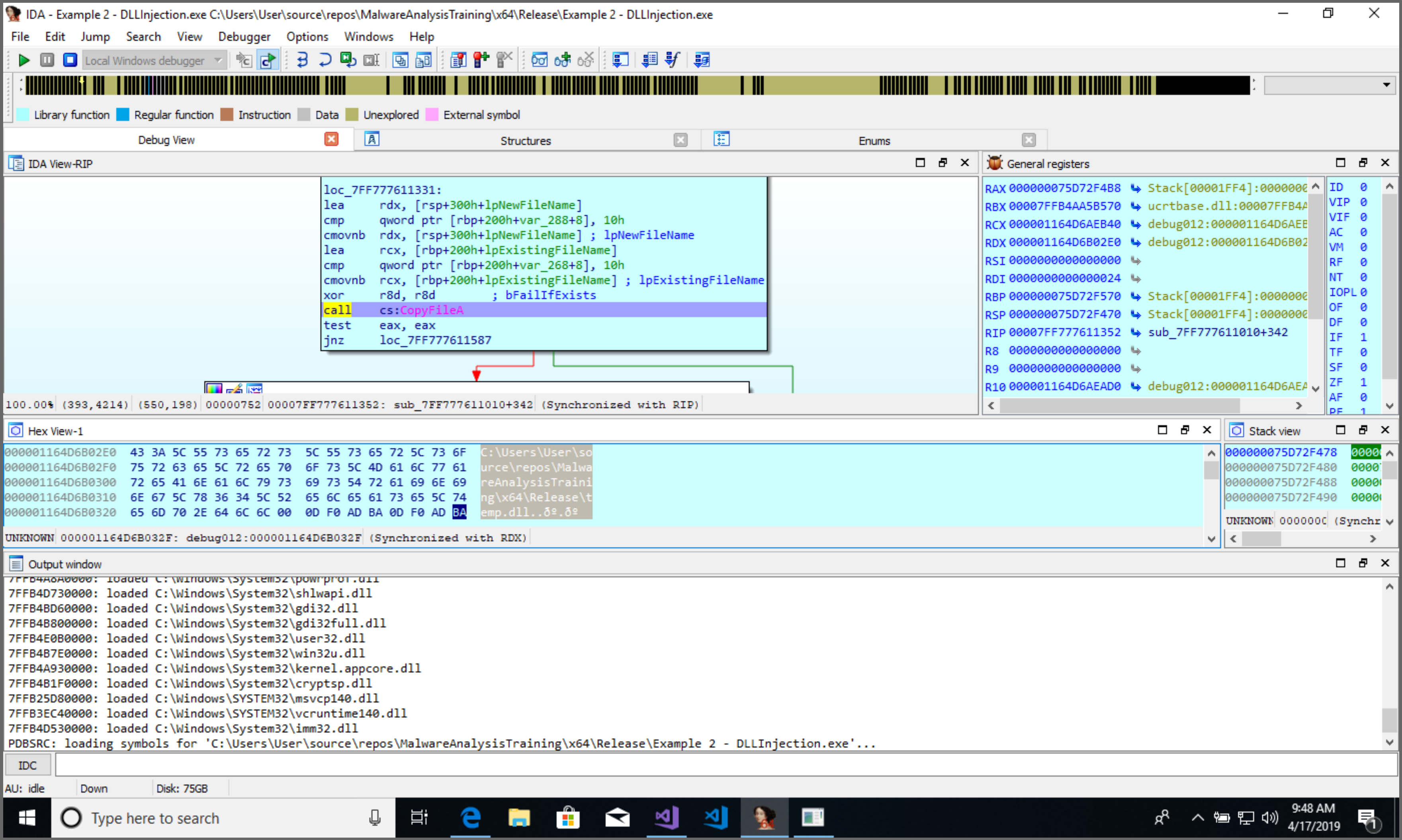
```
ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
IF   1
TF   0
SF   0
ZF   1
AF   0
PF   1
```

```
000001164D6AEB40  43 3A 5C 55 73 65 72 73  5C 55 73 65 72 5C 73 6F  C:\Users\User\so
000001164D6AEB50  75 72 63 65 5C 72 65 70  6F 73 5C 4D 61 6C 77 61  urce\repos\Malwa
000001164D6AEB60  72 65 41 6E 61 6C 79 73  69 73 54 72 61 69 6E 69  reAnalysisTraini
000001164D6AEB70  6E 67 5C 78 36 34 5C 52  65 6C 65 61 73 65 5C 69  ng\x64\Release\i
000001164D6AEB80  6E 6A 65 63 74 2E 64 6C  6C 00 AD BA 0D F0 AD BA  nject.dll.º.ðº
UNKNOWN 000001164D6AEB8F: debug012:000001164D6AEB8F (Synchronized with RCX)
```

```
000000075D72F478  0000
000000075D72F480  0000
000000075D72F488  0000
000000075D72F490  0000
UNKNOWN 0000000 (Synchr
```

```
7FFB4A8A0000: loaded C:\Windows\System32\powrprof.dll
7FFB4D730000: loaded C:\Windows\System32\shlwapi.dll
7FFB4BD60000: loaded C:\Windows\System32\gdi32.dll
7FFB4B800000: loaded C:\Windows\System32\gdi32full.dll
7FFB4E0B0000: loaded C:\Windows\System32\user32.dll
7FFB4B7E0000: loaded C:\Windows\System32\win32u.dll
7FFB4A930000: loaded C:\Windows\System32\kernel.appcore.dll
7FFB4B1F0000: loaded C:\Windows\System32\cryptsp.dll
7FFB25D80000: loaded C:\Windows\SYSTEM32\msvcp140.dll
7FFB3EC40000: loaded C:\Windows\SYSTEM32\vcruntime140.dll
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
```

Looks like it's named “temp.dll”. So what we know now is that the binary is basically copying the inject.dll and renaming it to temp.dll.



```
loc_7FF777611331:
lea     rdx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rdx, [rsp+300h+lpNewFileName] ; lpNewFileName
lea     rcx, [rbp+200h+lpExistingFileName]
cmp     qword ptr [rbp+200h+var_268+8], 10h
cmovnb  rcx, [rbp+200h+lpExistingFileName] ; lpExistingFileName
xor     r8d, r8d         ; bFailIfExists
call    cs:CopyFileA
test    eax, eax
jnz     loc_7FF777611587
```

```
100.00% (393,4214) (550,198) 00000752 00007FF777611352: sub_7FF777611010+342 (Synchronized with RIP)
```

```
RAX 000000075D72F4B8 ↳ Stack[00001FF4]:0000000
RBX 00007FFB4AA5B570 ↳ ucrtbase.dll:00007FFB4A
RCX 00001164D6AEB40 ↳ debug012:000001164D6AEE
RDX 00001164D6B02E0 ↳ debug012:000001164D6B02
RSI 0000000000000000 ↳
RDI 0000000000000024 ↳
RBP 000000075D72F570 ↳ Stack[00001FF4]:0000000
RSP 000000075D72F470 ↳ Stack[00001FF4]:0000000
RIP 00007FF777611352 ↳ sub_7FF777611010+342
R8  0000000000000000 ↳
R9  0000000000000000 ↳
R10 000001164D6AEAD0 ↳ debug012:000001164D6AEA
```

```
ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
IF   1
TF   0
SF   0
ZF   1
AF   0
PF   1
```

```
000001164D6B02E0  43 3A 5C 55 73 65 72 73  5C 55 73 65 72 5C 73 6F  C:\Users\User\so
000001164D6B02F0  75 72 63 65 5C 72 65 70  6F 73 5C 4D 61 6C 77 61  urce\repos\Malwa
000001164D6B0300  72 65 41 6E 61 6C 79 73  69 73 54 72 61 69 6E 69  reAnalysisTraini
000001164D6B0310  6E 67 5C 78 36 34 5C 52  65 6C 65 61 73 65 5C 74  ng\x64\Release\t
000001164D6B0320  65 6D 70 2E 64 6C 6C 00  0D F0 AD BA 0D F0 AD BA  emp.dll..ðº.ðº
UNKNOWN 000001164D6B032F: debug012:000001164D6B032F (Synchronized with RDX)
```

```
000000075D72F478  0000
000000075D72F480  0000
000000075D72F488  0000
000000075D72F490  0000
UNKNOWN 000000C (Synchr
```

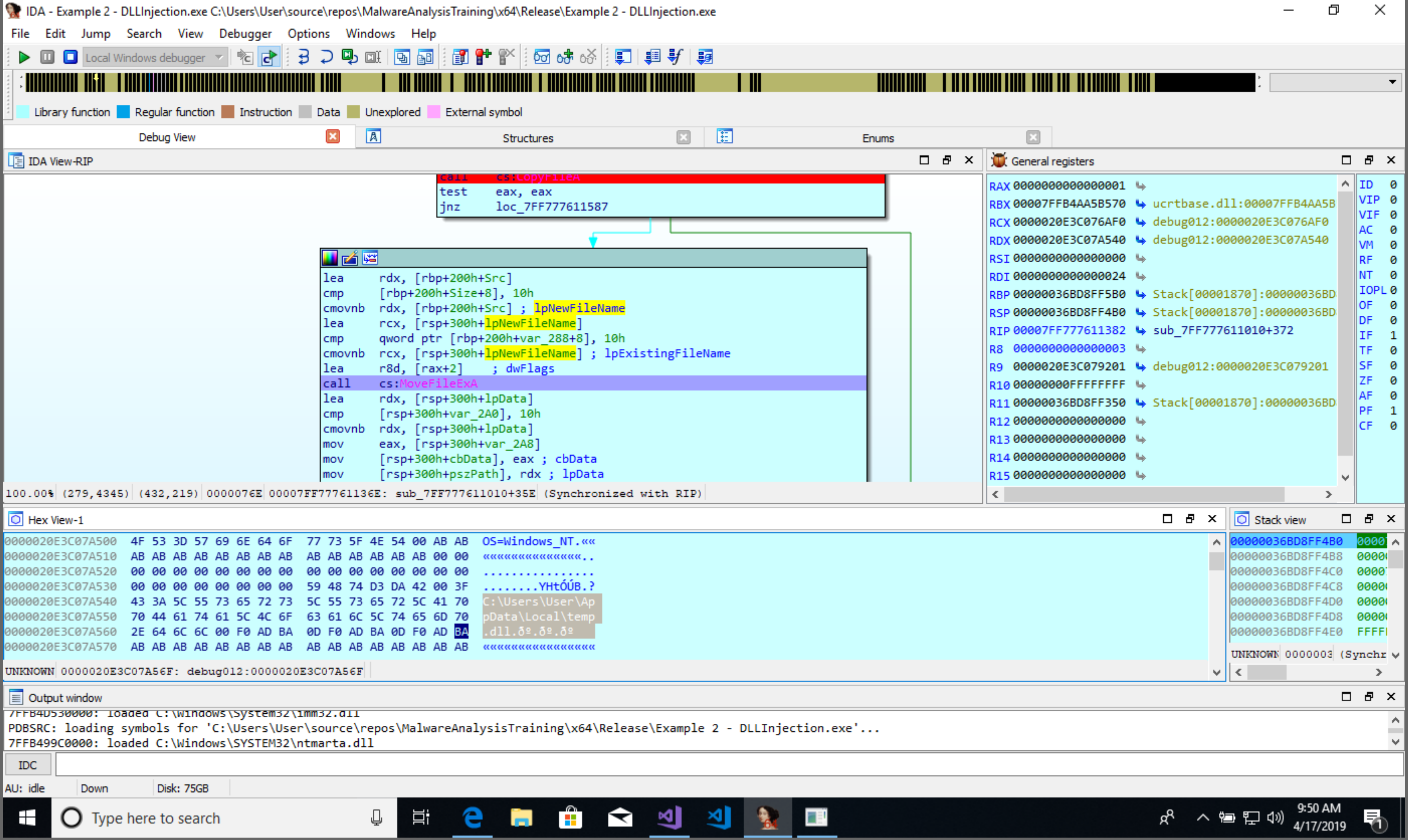
```
7FFB4A8A0000: loaded C:\Windows\System32\powrprof.dll
7FFB4D730000: loaded C:\Windows\System32\shlwapi.dll
7FFB4BD60000: loaded C:\Windows\System32\gdi32.dll
7FFB4B800000: loaded C:\Windows\System32\gdi32full.dll
7FFB4E0B0000: loaded C:\Windows\System32\user32.dll
7FFB4B7E0000: loaded C:\Windows\System32\win32u.dll
7FFB4A930000: loaded C:\Windows\System32\kernel.appcore.dll
7FFB4B1F0000: loaded C:\Windows\System32\cryptsp.dll
7FFB25D80000: loaded C:\Windows\SYSTEM32\msvcp140.dll
7FFB3EC40000: loaded C:\Windows\SYSTEM32\vcruntime140.dll
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
```

```
AU: idle   Down   Disk: 75GB
```

Next piece of evidence

- What about MoveFileExA?
- Let's put a breakpoint there and see what gets pushed through

Breakpoint and continuing to MoveFileExA and synchronizing with RDX shows us where the binary wishes to put the temp.dll. It looks like the file is being placed in AppData/Local.



IDA - Example 2 - DLLInjection.exe C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe

```
call    cs:CopyFileA
test    eax, eax
jnz     loc_7FF777611587

lea     rdx, [rbp+200h+Src]
cmp     [rbp+200h+Size+8], 10h
cmovnb  rdx, [rbp+200h+Src] ; lpNewFileName
lea     rcx, [rsp+300h+lpNewFileName]
cmp     qword ptr [rbp+200h+var_288+8], 10h
cmovnb  rcx, [rsp+300h+lpNewFileName] ; lpExistingFileName
lea     r8d, [rax+2]     ; dwFlags
call    cs:MoveFileExA
lea     rdx, [rsp+300h+lpData]
cmp     [rsp+300h+var_2A0], 10h
cmovnb  rdx, [rsp+300h+lpData]
mov     eax, [rsp+300h+var_2A8]
mov     [rsp+300h+cbData], eax ; cbData
mov     [rsp+300h+pszPath], rdx ; lpData
```

```
100.00% (279,4345) (432,219) 0000076E 00007FF77761136E: sub_7FF777611010+35E (Synchronized with RIP)
```

```
RAX 0000000000000001
RBX 00007FFB4AA5B570  ucrtbase.dll:00007FFB4AA5B
RCX 0000020E3C076AF0  debug012:0000020E3C076AF0
RDX 0000020E3C07A540  debug012:0000020E3C07A540
RSI 0000000000000000
RDI 0000000000000024
RBP 00000036BD8FF5B0  Stack[00001870]:00000036BD
RSP 00000036BD8FF4B0  Stack[00001870]:00000036BD
RIP 00007FF777611382  sub_7FF777611010+372
R8  0000000000000003
R9  0000020E3C079201  debug012:0000020E3C079201
R10 00000000FFFFFFFF
R11 00000036BD8FF350  Stack[00001870]:00000036BD
R12 0000000000000000
R13 0000000000000000
R14 0000000000000000
R15 0000000000000000

ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
IF   1
TF   0
SF   0
ZF   0
AF   0
PF   1
CF   0
```

```
0000020E3C07A500  4F 53 3D 57 69 6E 64 6F  77 73 5F 4E 54 00 AB AB  OS=Windows_NT.««
0000020E3C07A510  AB AB AB AB AB AB AB AB  AB AB AB AB AB AB 00 00  ««««««««««««««..
0000020E3C07A520  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
0000020E3C07A530  00 00 00 00 00 00 00 00  59 48 74 D3 DA 42 00 3F  ........YHtÓÚB.?
0000020E3C07A540  43 3A 5C 55 73 65 72 73  5C 55 73 65 72 5C 41 70  C:\Users\User\Ap
0000020E3C07A550  70 44 61 74 61 5C 4C 6F  63 61 6C 5C 74 65 6D 70  pData\Local\temp
0000020E3C07A560  2E 64 6C 6C 00 F0 AD BA  0D F0 AD BA 0D F0 AD BA  .dll.ðº.ðº.ðº
0000020E3C07A570  AB AB AB AB AB AB AB AB  AB AB AB AB AB AB AB AB  ««««««««««««««««
UNKNOWN 0000020E3C07A56F: debug012:0000020E3C07A56F
```

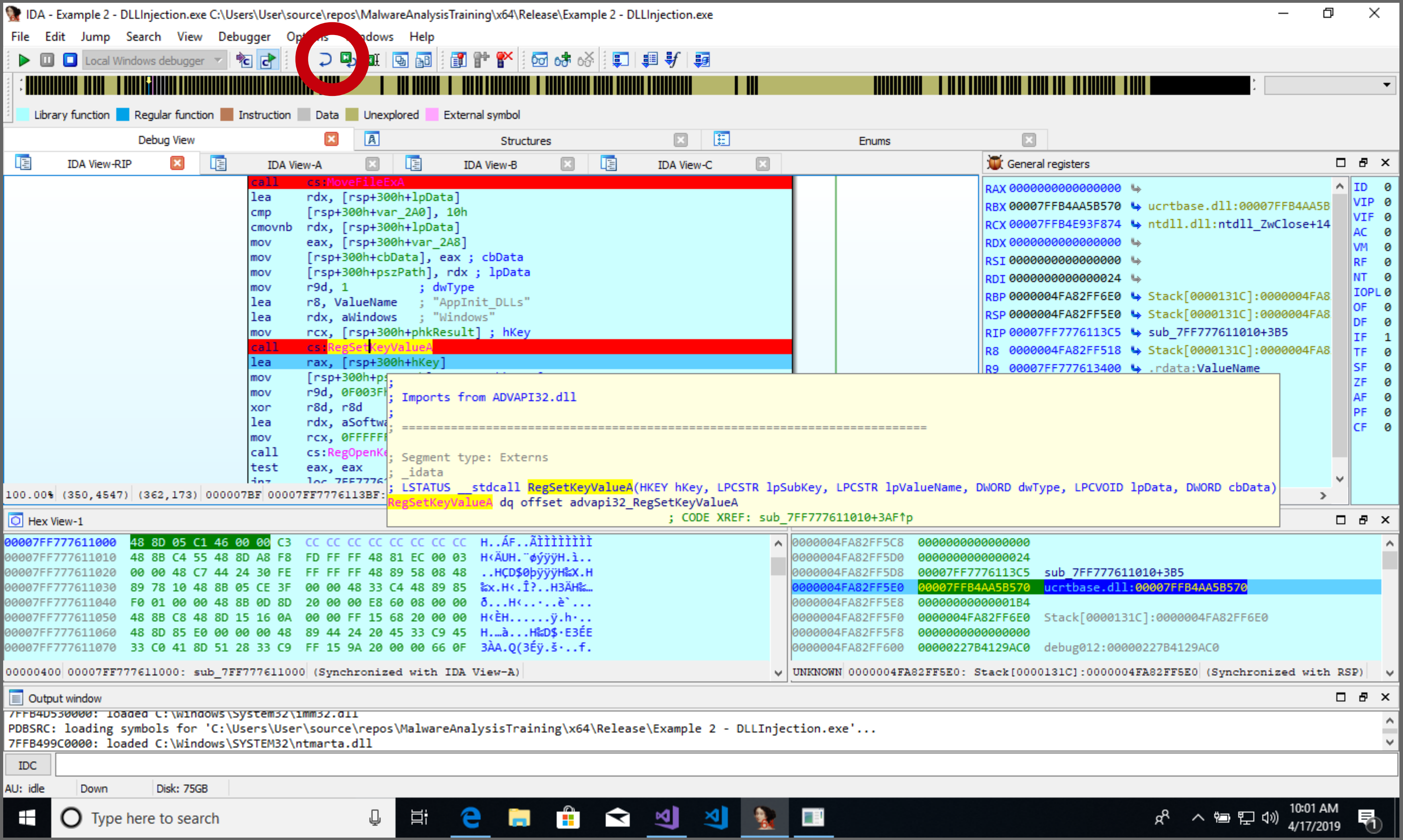
```
00000036BD8FF4B0  0000
00000036BD8FF4B8  0000
00000036BD8FF4C0  0000
00000036BD8FF4C8  0000
00000036BD8FF4D0  0000
00000036BD8FF4D8  0000
00000036BD8FF4E0  FFFF
UNKNOWN 0000003 (Synchr
```

```
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
7FFB499C0000: loaded C:\Windows\SYSTEM32\ntmarta.dll
```

AU: idle   Down   Disk: 75GB

9:50 AM
4/17/2019

Now lets breakpoint on RegSetValueA and step one line over it. I circled in red below the jump over button. Let's go into “regedit” and check the registry to see if the binary did modify AppInit_DLLs



```
call    cs:MoveFileExA
lea     rdx, [rsp+300h+lpData]
cmp     [rsp+300h+var_2A0], 10h
cmovnb  rdx, [rsp+300h+lpData]
mov     eax, [rsp+300h+var_2A8]
mov     [rsp+300h+cbData], eax ; cbData
mov     [rsp+300h+pszPath], rdx ; lpData
mov     r9d, 1          ; dwType
lea     r8, ValueName   ; "AppInit_DLLs"
lea     rdx, aWindows   ; "Windows"
mov     rcx, [rsp+300h+phkResult] ; hKey
call    cs:RegSetKeyValueA
lea     rax, [rsp+300h+hKey]
mov     [rsp+300h+ps
mov     r9d, 0F003Fh
xor     r8d, r8d
lea     rdx, aSoftwa
mov     rcx, 0FFFFFF
call    cs:RegOpenKe
test    eax, eax
jnz     loc_7FF7776
100.00% (350,4547) (362,173) 000007BF 00007FF7776113BF:
```

```
;
; Imports from ADVAPI32.dll
;
; ===============================================================================

; Segment type: Externs
; _idata
; LSTATUS __stdcall RegSetKeyValueA(HKEY hKey, LPCSTR lpSubKey, LPCSTR lpValueName, DWORD dwType, LPCVOID lpData, DWORD cbData)
RegSetKeyValueA dq offset advapi32_RegSetKeyValueA
                                        ; CODE XREF: sub_7FF777611010+3AF↑p
```

```
RAX 0000000000000000 ↳
RBX 00007FFB4AA5B570 ↳ ucrtbase.dll:00007FFB4AA5B
RCX 00007FFB4E93F874 ↳ ntdll.dll:ntdll_ZwClose+14
RDX 0000000000000000 ↳
RSI 0000000000000000 ↳
RDI 0000000000000024 ↳
RBP 0000004FA82FF6E0 ↳ Stack[0000131C]:0000004FA8
RSP 0000004FA82FF5E0 ↳ Stack[0000131C]:0000004FA8
RIP 00007FF7776113C5 ↳ sub_7FF777611010+3B5
R8  0000004FA82FF518 ↳ Stack[0000131C]:0000004FA8
R9  00007FF777613400 ↳ .rdata:ValueName
```

```
ID   0
VIP  0
VIF  0
AC   0
VM   0
RF   0
NT   0
IOPL 0
OF   0
DF   0
IF   1
TF   0
SF   0
ZF   0
AF   0
PF   0
CF   0
```

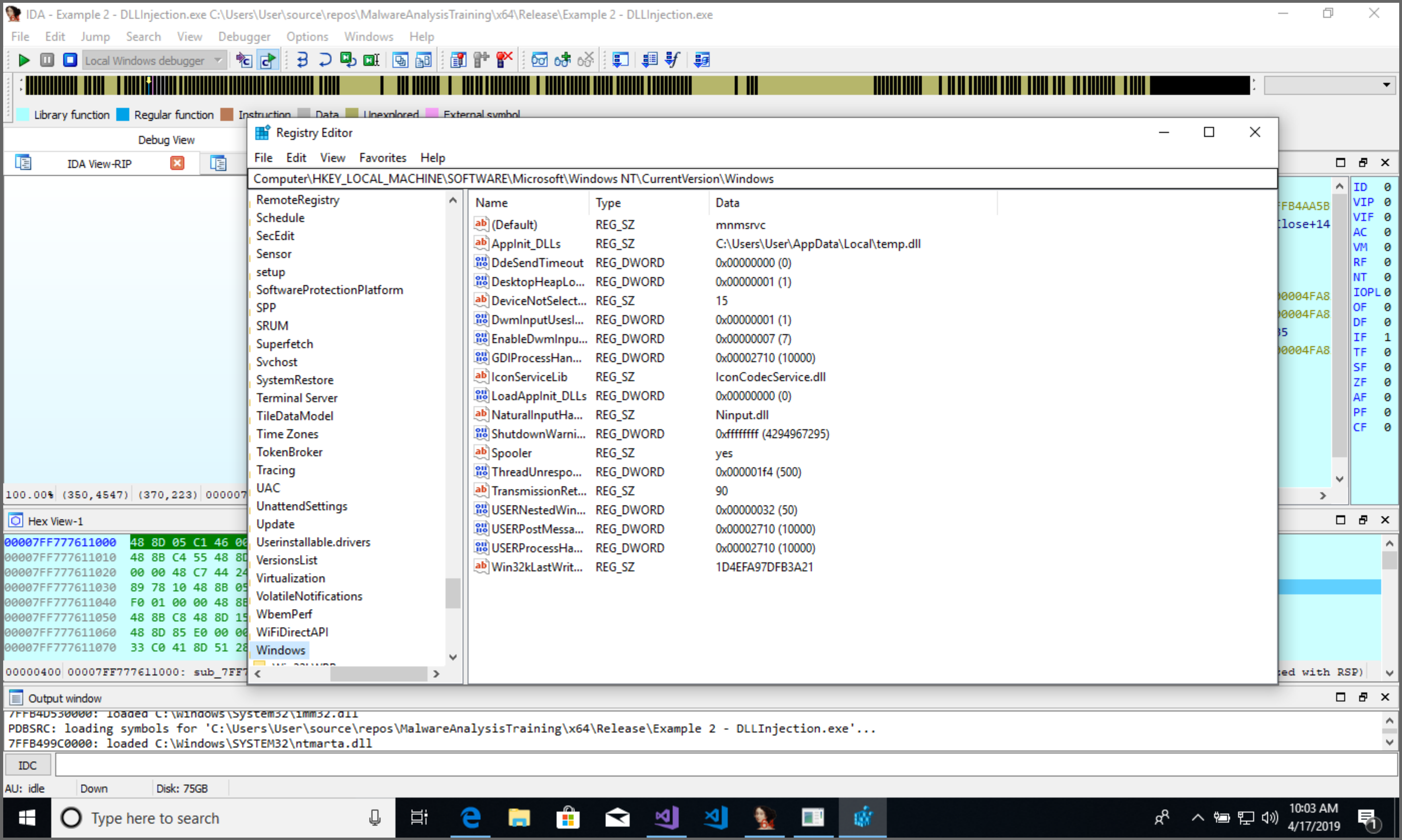
```
00007FF777611000  48 8D 05 C1 46 00 00 C3  CC CC CC CC CC CC CC CC  H..ÁF..ÃÌÌÌÌÌÌÌÌ
00007FF777611010  48 8B C4 55 48 8D A8 F8  FD FF FF 48 81 EC 00 03  H‹ÄUH.¨øýÿÿH.ì..
00007FF777611020  00 00 48 C7 44 24 30 FE  FF FF FF 48 89 58 08 48  ..HÇD$0þÿÿÿH‰X.H
00007FF777611030  89 78 10 48 8B 05 CE 3F  00 00 48 33 C4 48 89 85  ‰x.H‹.Î?..H3ÄH‰…
00007FF777611040  F0 01 00 00 48 8B 0D 8D  20 00 00 E8 60 08 00 00  ð...H‹..·..è`...
00007FF777611050  48 8B C8 48 8D 15 16 0A  00 00 FF 15 68 20 00 00  H‹ÈH......ÿ.h·..
00007FF777611060  48 8D 85 E0 00 00 00 48  89 44 24 20 45 33 C9 45  H.…à...H‰D$·E3ÉE
00007FF777611070  33 C0 41 8D 51 28 33 C9  FF 15 9A 20 00 00 66 0F  3ÀA.Q(3Éÿ.š·..f.
00000400 00007FF777611000: sub_7FF777611000 (Synchronized with IDA View-A)
```

```
0000004FA82FF5C8  0000000000000000
0000004FA82FF5D0  0000000000000024
0000004FA82FF5D8  00007FF7776113C5  sub_7FF777611010+3B5
0000004FA82FF5E0  00007FFB4AA5B570  ucrtbase.dll:00007FFB4AA5B570
0000004FA82FF5E8  00000000000001B4
0000004FA82FF5F0  0000004FA82FF6E0  Stack[0000131C]:0000004FA82FF6E0
0000004FA82FF5F8  0000000000000000
0000004FA82FF600  00000227B4129AC0  debug012:00000227B4129AC0
UNKNOWN 0000004FA82FF5E0: Stack[0000131C]:0000004FA82FF5E0 (Synchronized with RSP)
```

```
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
7FFB499C0000: loaded C:\Windows\SYSTEM32\ntmarta.dll
```

AU: idle   Down   Disk: 75GB

That looks like a yes to me. The binary replaced the AppInit_DLLs with the temp.dll hiding in AppData/Local. Also if you noticed LoadInit_DLLs is now the value “1”.



IDA - Example 2 - DLLInjection.exe C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe

Registry Editor

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows

RemoteRegistry
Schedule
SecEdit
Sensor
setup
SoftwareProtectionPlatform
SPP
SRUM
Superfetch
Svchost
SystemRestore
Terminal Server
TileDataModel
Time Zones
TokenBroker
Tracing
UAC
UnattendSettings
Update
Userinstallable.drivers
VersionsList
Virtualization
VolatileNotifications
WbemPerf
WiFiDirectAPI
Windows

Name	Type	Data
(Default)	REG_SZ	mnmsrvc
AppInit_DLLs	REG_SZ	C:\Users\User\AppData\Local\temp.dll
DdeSendTimeout	REG_DWORD	0x00000000 (0)
DesktopHeapLo...	REG_DWORD	0x00000001 (1)
DeviceNotSelect...	REG_SZ	15
DwmInputUsesI...	REG_DWORD	0x00000001 (1)
EnableDwmInpu...	REG_DWORD	0x00000007 (7)
GDIProcessHan...	REG_DWORD	0x00002710 (10000)
IconServiceLib	REG_SZ	IconCodecService.dll
LoadAppInit_DLLs	REG_DWORD	0x00000000 (0)
NaturalInputHa...	REG_SZ	Ninput.dll
ShutdownWarni...	REG_DWORD	0xffffffff (4294967295)
Spooler	REG_SZ	yes
ThreadUnrespo...	REG_DWORD	0x000001f4 (500)
TransmissionRet...	REG_SZ	90
USERNestedWin...	REG_DWORD	0x00000032 (50)
USERPostMessa...	REG_DWORD	0x00002710 (10000)
USERProcessHa...	REG_DWORD	0x00002710 (10000)
Win32kLastWrit...	REG_SZ	1D4EFA97DFB3A21

```
100.00% (350,4547) (370,223) 000007
```

```
00007FF777611000  48 8D 05 C1 46 00
00007FF777611010  48 8B C4 55 48 8D
00007FF777611020  00 00 48 C7 44 24
00007FF777611030  89 78 10 48 8B 05
00007FF777611040  F0 01 00 00 48 8B
00007FF777611050  48 8B C8 48 8D 15
00007FF777611060  48 8D 85 E0 00 00
00007FF777611070  33 C0 41 8D 51 28
00000400 00007FF777611000: sub_7FF7
```

```
7FFB4D530000: loaded C:\Windows\System32\imm32.dll
PDBSRC: loading symbols for 'C:\Users\User\source\repos\MalwareAnalysisTraining\x64\Release\Example 2 - DLLInjection.exe'...
7FFB499C0000: loaded C:\Windows\SYSTEM32\ntmarta.dll
```

Current analysis

- So we have the general idea of what's happening with the main binary.
- It's copying/renaming the inject.dll to temp.dll, moving temp.dll to AppData/Local.
- Then modifying the AppInit_DLLs registry to point to that “.dll”.

Further analysis

- Now using the normal analysis routine of static analysis and dynamic analysis. Perform an analysis of the hidden “temp.dll” in the AppData/Local directory and see what you find.
- What does the temp.dll do?
- Is it also malicious?

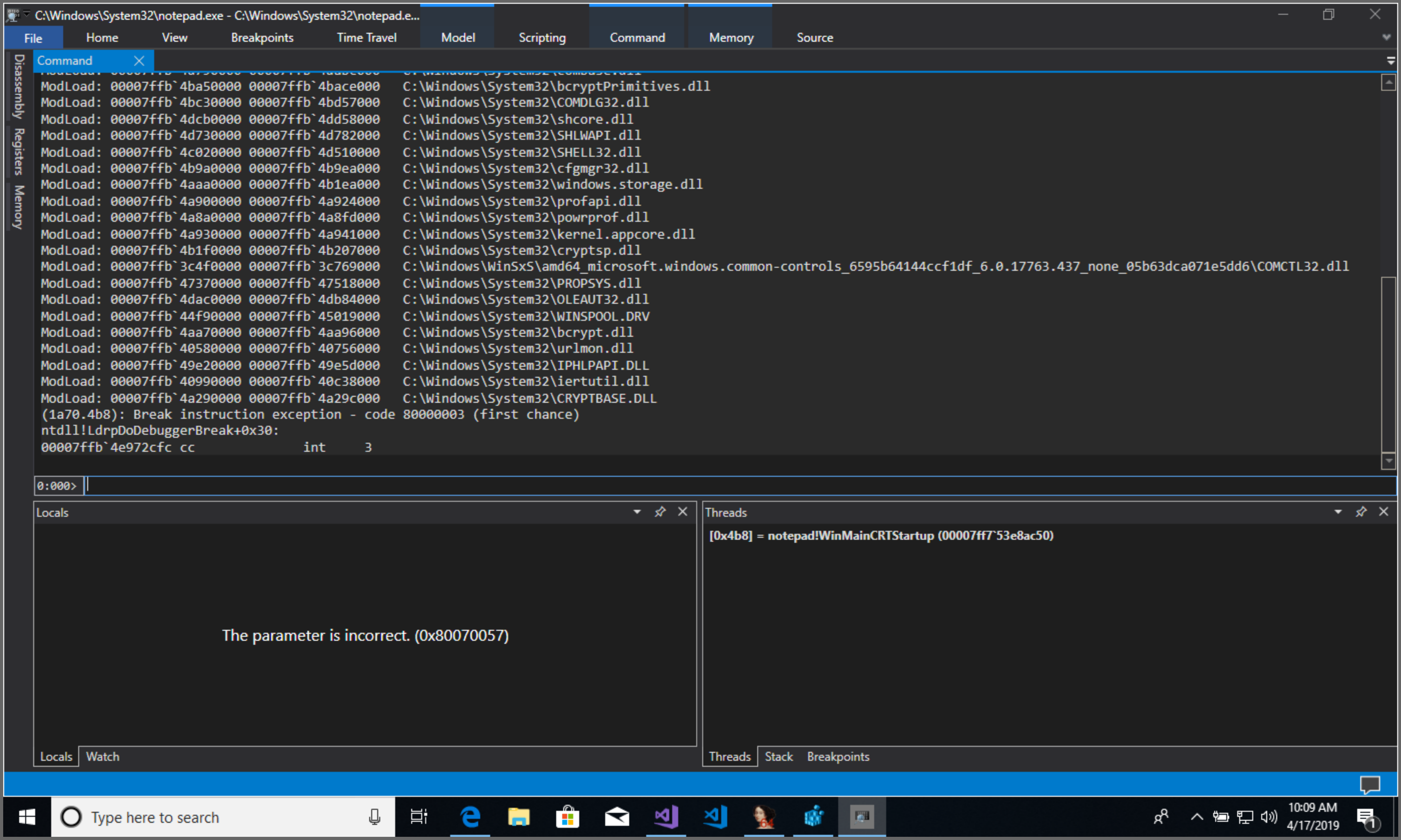
Further analysis cont.

- The answer is yes. It is malicious.
- Why?
- Was it a bit trickier to figure out what this “.dll” is meant to do?

Further analysis cont.

- The way I proved its maliciousness was using Windbg to prove my answer.
- What I did was open notepad, which uses user32.dll, in windbg and watched what happens.

Let's open windbg and launch notepad.exe within windbg. Then hit “GO” and watch happens.



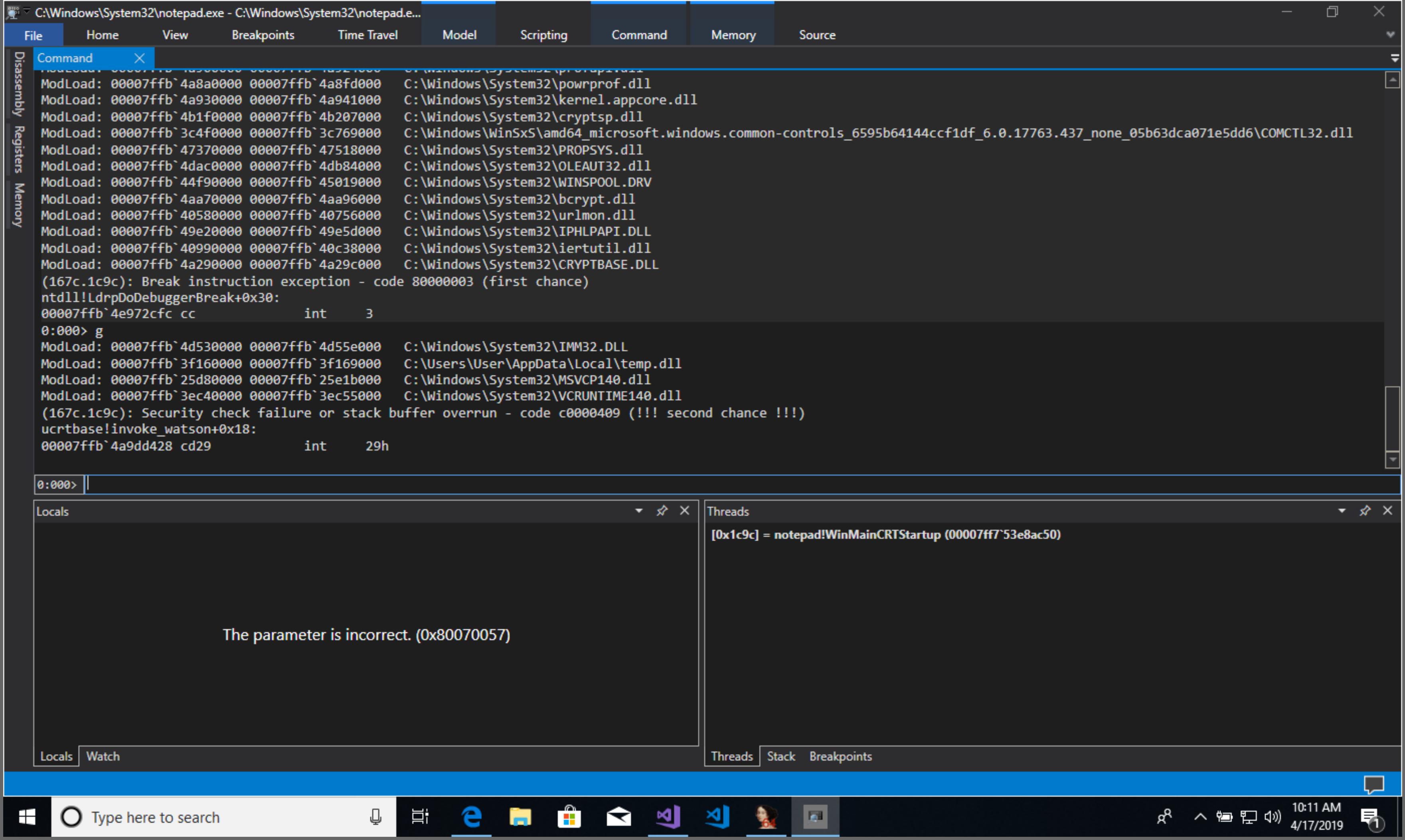
```
ModLoad: 00007ffb`4ba50000 00007ffb`4bace000   C:\Windows\System32\bcryptPrimitives.dll
ModLoad: 00007ffb`4bc30000 00007ffb`4bd57000   C:\Windows\System32\COMDLG32.dll
ModLoad: 00007ffb`4dcb0000 00007ffb`4dd58000   C:\Windows\System32\shcore.dll
ModLoad: 00007ffb`4d730000 00007ffb`4d782000   C:\Windows\System32\SHLWAPI.dll
ModLoad: 00007ffb`4c020000 00007ffb`4d510000   C:\Windows\System32\SHELL32.dll
ModLoad: 00007ffb`4b9a0000 00007ffb`4b9ea000   C:\Windows\System32\cfgmgr32.dll
ModLoad: 00007ffb`4aaa0000 00007ffb`4b1ea000   C:\Windows\System32\windows.storage.dll
ModLoad: 00007ffb`4a900000 00007ffb`4a924000   C:\Windows\System32\profapi.dll
ModLoad: 00007ffb`4a8a0000 00007ffb`4a8fd000   C:\Windows\System32\powrprof.dll
ModLoad: 00007ffb`4a930000 00007ffb`4a941000   C:\Windows\System32\kernel.appcore.dll
ModLoad: 00007ffb`4b1f0000 00007ffb`4b207000   C:\Windows\System32\cryptsp.dll
ModLoad: 00007ffb`3c4f0000 00007ffb`3c769000   C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.17763.437_none_05b63dca071e5dd6\COMCTL32.dll
ModLoad: 00007ffb`47370000 00007ffb`47518000   C:\Windows\System32\PROPSYS.dll
ModLoad: 00007ffb`4dac0000 00007ffb`4db84000   C:\Windows\System32\OLEAUT32.dll
ModLoad: 00007ffb`44f90000 00007ffb`45019000   C:\Windows\System32\WINSPOOL.DRV
ModLoad: 00007ffb`4aa70000 00007ffb`4aa96000   C:\Windows\System32\bcrypt.dll
ModLoad: 00007ffb`40580000 00007ffb`40756000   C:\Windows\System32\urlmon.dll
ModLoad: 00007ffb`49e20000 00007ffb`49e5d000   C:\Windows\System32\IPHLPAPI.DLL
ModLoad: 00007ffb`40990000 00007ffb`40c38000   C:\Windows\System32\iertutil.dll
ModLoad: 00007ffb`4a290000 00007ffb`4a29c000   C:\Windows\System32\CRYPTBASE.DLL
(1a70.4b8): Break instruction exception - code 80000003 (first chance)
ntdll!LdrpDoDebuggerBreak+0x30:
00007ffb`4e972cfc cc              int     3
0:000>
```

The parameter is incorrect. (0x80070057)

[0x4b8] = notepad!WinMainCRTStartup (00007ff7`53e8ac50)

So what does this mean?

It depends on the context. Most malware doesn't want anti-virus popping up. Anti-virus uses user32 to render messages on your screen. So this .dll in particular prevents such an event by crashing anything that loads user32.dll



```
ModLoad: 00007ffb`4a8a0000 00007ffb`4a8fd000   C:\Windows\System32\powrprof.dll
ModLoad: 00007ffb`4a930000 00007ffb`4a941000   C:\Windows\System32\kernel.appcore.dll
ModLoad: 00007ffb`4b1f0000 00007ffb`4b207000   C:\Windows\System32\cryptsp.dll
ModLoad: 00007ffb`3c4f0000 00007ffb`3c769000   C:\Windows\WinSxS\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.17763.437_none_05b63dca071e5dd6\COMCTL32.dll
ModLoad: 00007ffb`47370000 00007ffb`47518000   C:\Windows\System32\PROPSYS.dll
ModLoad: 00007ffb`4dac0000 00007ffb`4db84000   C:\Windows\System32\OLEAUT32.dll
ModLoad: 00007ffb`44f90000 00007ffb`45019000   C:\Windows\System32\WINSPOOL.DRV
ModLoad: 00007ffb`4aa70000 00007ffb`4aa96000   C:\Windows\System32\bcrypt.dll
ModLoad: 00007ffb`40580000 00007ffb`40756000   C:\Windows\System32\urlmon.dll
ModLoad: 00007ffb`49e20000 00007ffb`49e5d000   C:\Windows\System32\IPHLPAPI.DLL
ModLoad: 00007ffb`40990000 00007ffb`40c38000   C:\Windows\System32\iertutil.dll
ModLoad: 00007ffb`4a290000 00007ffb`4a29c000   C:\Windows\System32\CRYPTBASE.DLL
(167c.1c9c): Break instruction exception - code 80000003 (first chance)
ntdll!LdrpDoDebuggerBreak+0x30:
00007ffb`4e972cfc cc              int     3
0:000> g
ModLoad: 00007ffb`4d530000 00007ffb`4d55e000   C:\Windows\System32\IMM32.DLL
ModLoad: 00007ffb`3f160000 00007ffb`3f169000   C:\Users\User\AppData\Local\temp.dll
ModLoad: 00007ffb`25d80000 00007ffb`25e1b000   C:\Windows\System32\MSVCP140.dll
ModLoad: 00007ffb`3ec40000 00007ffb`3ec55000   C:\Windows\System32\VCRUNTIME140.dll
(167c.1c9c): Security check failure or stack buffer overrun - code c0000409 (!!! second chance !!!)
ucrtbase!invoke_watson+0x18:
00007ffb`4a9dd428 cd29            int     29h

0:000>
```

The parameter is incorrect. (0x80070057)

[0x1c9c] = notepad!WinMainCRTStartup (00007ff7`53e8ac50)

Closing the Analysis

- So we've developed a pretty reasonable timeline at this point.
- The registry key of AppInit_DLLs is being used by the malware to load a “temp.dll” that prevents anything loading user32.dll from running.
- This is a pretty simple summary, but details the analysis perfectly.
- We can consider our analysis finished here.